Design and Evaluation of Online Fault Diagnosis Protocols for Wireless Networks

Manmath Narayan Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

Design and Evaluation of Online Fault Diagnosis Protocols for Wireless Networks

Thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Manmath Narayan Sahoo
(Roll: 509CS404)

under the supervision of

Prof. Pabitra Mohan Khilar
NIT Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India
June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India.

Dr. Pabitra Mohan Khilar
Assistant Professor

Dec 06, 2014

Certificate

This is to certify that the work in the thesis entitled ***Design and Evaluation of Online Fault Diagnosis Protocols for Wireless Networks*** by ***Manmath Narayan Sahoo*** is a record of an original research work carried out under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Doctor in Philosophy in Computer Science and Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Pabitra Mohan Khilar

Acknowledgment

"None of us got to where we are alone. Whether the assistance we received was obvious or subtle, acknowledging someone's help is a big part of understanding the importance of saying thank you."

–Harvey Mackay

Before presenting the thesis work, I would like to mention a few words for the individuals who gave their unending help and support in completing this thesis.

With my sincere gratitude, I acknowledge the guidance, support, and constant encouragement rendered by my supervisor, Prof. Pabitra Mohan Khilar during the course of my work. It would have been difficult to complete this thesis without his endless enthusiasm, knowledge, patience, and answers to my numerous questions. I am indebted to Dr. S. K. Jena, Dr. S. K. Rath, Dr. B. Majhi, Dr. A. K. Turuk, Dr. D. P. Mohapatra, Dr. R. Baliarsingh, Dr. B. D. Sahoo for their insightful feedback and encouragement which helped me to improve the presentation of the thesis in many ways. I am deeply grateful to Prof. G. K. Panda for his advice and meticulous comments that were an enormous help to me.

I also owe a great debt of gratitude to all the members of the Department of Computer Science and Engineering, and the Institute for providing me with the necessary facilities and assistance. Not to forget, many thanks to the wonderful reviewers of my articles for their constructive comments on the work.

Probably, thanks would not be enough to convey my love and gratitude to my family and friends who have been a constant source of motivation, support, and concern throughout.

Manmath Narayan Sahoo

Abstract

Any node in a network, or a component of it may fail and show undesirable behavior due to physical defects, imperfections, or hardware and/or software related glitches. Presence of faulty hosts in the network affects the computational efficiency, and quality of service (QoS). This calls for the development of efficient fault diagnosis protocols to detect and handle faulty hosts. Fault diagnosis protocols designed for wired networks cannot directly be propagated to wireless networks, due to difference in characteristics, and requirements. This thesis work unravels system level fault diagnosis protocols for wireless networks, particularly for Mobile ad hoc Networks (MANETs), and Wireless Sensor Networks (WSNs), considering faults based on their persistence (permanent, intermittent, and transient), and node mobility.

Based on the comparisons of outcomes of the same tasks (*comparison model*), a distributed diagnosis protocol has been proposed for static topology MANETs, where a node requires to respond to only one test request from its neighbors, that reduces the communication complexity of the diagnosis process. A novel approach to handle more intractable intermittent faults in dynamic topology MANETs is also discussed. Based on the spatial correlation of sensor measurements, a distributed fault diagnosis protocol is developed to classify the nodes to be fault-free, permanently faulty, or intermittently faulty, in WSNs. The nodes affected by transient faults are often considered fault-free, and should not be isolated from the network. Keeping this objective in mind, we have developed a diagnosis algorithm for WSNs to discriminate transient faults from intermittent and permanent faults.

After each node finds the status of all 1-hop neighbors (*local diagnostic view*), these views are disseminated among the fault-free nodes to deduce the fault status of all nodes in the network (*global diagnostic view*). A spanning tree based dissemination strategy is adopted, instead of conventional flooding, to have less communication complexity. Analytically, the proposed protocols are shown to be correct, and complete. The protocols are implemented using INET-20111118 (for MANETs) and Castalia-3.2 (for WSNs) on OMNeT++ 4.2 platform. The obtained simulation results for accuracy and false alarm rate vouch the feasibility and efficiency of the proposed algorithms over existing landmark protocols.

Keywords: MANETs, WSNs, fault diagnosis, comparison model, information dissemination.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ARE Average number of Rounds for Exclusion

ARR Average number of Rounds for Reintegration

CA Classification Accuracy

CH Cluster Head

DA Detection Accuracy

DCH Deputy Cluster Head

DSDP Distributed Self Diagnosis Protocol

FAR False Alarm Rate

FDP Fault Diagnosis Protocol

FIR Fault Inspection and Recovery

MANET Mobile Ad hoc Network

QoS Quality of Service

ST Spanning Tree

WSN Wireless Sensor Network

Chapter 1

Introduction

Moore's law, a commonly used theory in semiconductor industry states that the number of transistors on integrated circuits doubles approximately every eighteen months. The apparent consequence of this law is the ability to embed number of cores on the same chip. This made us realize high-speed computation. Besides, it can also be observed that the integrated circuits have become compact, less expensive, and more importantly, less power hungry. Thus, designing a large computer network is no more becoming a nightmare. However, the complexity of the network increases with the increase in network size. Such complex networks work constantly serving specific goals, making our lives more pleasant and comfortable until they are affected by failures.

Any node (*hosts*, and *units* can interchangeably be used for *nodes*), or a component of it may fail at any time; leading the application running on that node to be defunct, or to produce incorrect results. So the presence of faulty units may degrade the system performance by corrupting the network. As severe consequences, faults in a network may lead to human, economic, and environmental loses; especially in case of safety-critical applications. In such complex network systems, occurrences of faults cannot be prevented. However, the consequences of faults can be avoided, or at least their severity could be minimized by properly handling the faulty units. Moreover, the whole system is not considered ineffectual, if some of the nodes in the network become faulty. These give rise to the requirements for the development of fault diagnosis protocols. The objective of a Fault Diagnosis Protocol (FDP) is to detect the faulty events in the network, and let all fault-free units to receive these faulty events; thus making the network still operational in the presence of faults, but of course with

degraded performance. Essentially, fault diagnosis is a process of segregating the set of faulty nodes from the set of fault-free ones, and to provide each fault-free node with a global diagnostic view comprising the status of every node in the system. In other words, FDP is intended to draw a consensus among the fault-free units about the status of all faulty units in the system. The FDPs provide a convenient and vital panorama of the condition of the network. Since the last few decades, fault diagnosis has been a focussed area of research due to its wide range of applications that range from small local area networks to large scale real time systems.

The terminologies in this field are not consistent. This can be sighted from the simple questions, *viz.*, what are the differences between faults, errors, malfunctions, and failures? How fault identification, detection, isolation, and diagnosis are different? These inconsistencies make it difficult to understand the objectives of the contributions and hence, to compare various approaches [1]. In Section 1.1, various terminologies in the field of fault diagnosis are enlightened.

1.1 Faults and Remedies

Faults are abnormal states of a node caused due to various factors such as temperature, design and installation errors, age, power depletion, etc. Errors in computation and/or communication arise, when fault affects a node or link in the network. It is important to note that many errors do not cause a failure. A fault is *active* when it causes an error, otherwise it is *dormant* [2]. Malfunctions and failures; both are the manifestation of errors in a system that makes the system components unable to work properly. Nevertheless, if errors occur in system components intermittently, then the system is said to be malfunctioning, but if components show error continuously then it results in system failure. Some commonly accepted definitions of these terms are as follows:

(i) Fault: An unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable, usual, or standard condition.

(ii) Error: A deviation between a measured or computed value (of an output variable) and the true, specified, or theoretically correct value.

(iii) Malfunction: An intermittent irregularity in the fulfilment of a system's desired function.

(iv) Failure: A permanent interruption of a system's ability to perform a required function under specified operating conditions.

A fault diagnosis system that monitors the nodes in the network for possible faults, often performs the following tasks:

(i) Fault detection: Determining if faults are present in a system.

(ii) Fault isolation: Determining the location of a fault. Follows fault detection.

(iii) Fault identification: Determining the kind and the behavior of a fault. Follows fault isolation.

(iv) Fault diagnosis: Determining the kind, and location of a fault. Follows fault detection. Includes fault isolation and identification.

Fault detection results in a binary decision if a system is affected by faults or not. Fault isolation aims at determining which of the nodes/components are faulty, but does not concentrate on the types of faults and their behavior, which are the sole objectives of fault identification. Essentially, fault diagnosis covers both fault isolation, and identification. It provides a consistent global view of the fault status of the whole system to each node in the network.

1.1.1 Fault Modelling and Classification

A fault model describes the behavior of a node that is affected by faults. Modelling of faults in a node can be studied at different level of abstractions such as software level, hardware level, or node/system level. A system level fault may result from hardware or software related glitches. In this thesis work, we concentrate on system level faults without moving into much detail of hardware or software related flaws.

FDPs are designed to handle various types of faults. Faults can be categorized based on their persistence, the underlying cause(s), or the behavior of the faulty components. Based on the duration of persistence, they can be of three types: permanent,

intermittent, or transient. Permanent faults are characterized by their continuous faulty behavior, and the only way to handle such faults is to repair or replace the faulty component, or the node as a whole. In contrast, intermittent, and transient faults are temporary in nature. Intermittent faults are caused due to some internal components, showing unusual behavior. After their first appearance, they exhibit high recurrence rate, and eventually tend to become permanent [3, 4]. On the other hand, transient faults are often caused due to external errors (e.g., noise), and their adverse effects disappear rapidly. Due to the unpredictable behaviors of transient and intermittent faults, they are hard to diagnose and handle as compared to permanent faults. The behavior of these faults in time domain is clearly shown in Figure 1.1.

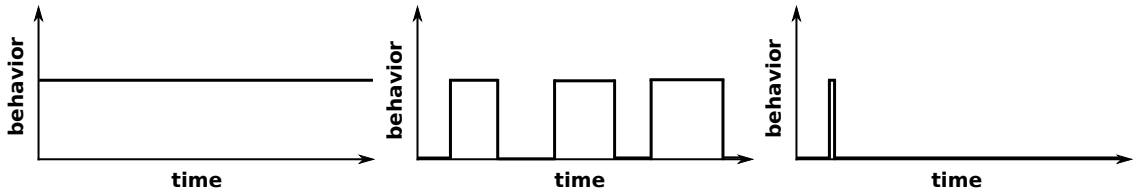


Figure 1.1: Behavior of (a) Permanent (b) Intermittent, and (c) Transient faults

Nodes in a faulty state, may or may not be able to communicate. If they are neither able to perform any computation nor to communicate with their neighbors, then they are said to be *hard faulty (crash faulty)*; otherwise, if they communicate with altered behaviors with their neighbors and the computational results are not correct, then they are said to be *soft faulty (value faulty)*. Faults can also be categorized based on whether they are allowed to be induced during the diagnosis session or not. Under this consideration, faults are of two types: *static*, and *dynamic*. If new faults are allowed to be induced once the diagnosis session has been started, then the faults are of dynamic type; otherwise they are termed as static faults.

A more comprehensive classification of faults has been studied in [3], where faults are categorized into the following classes: fail-stop faults, crash faults, omission faults, timing faults, incorrect computation faults, and Byzantine faults. Knowledge of all possible fault classes not only allows a protocol developer to develop generic diagnosis protocols, also makes it possible to compare the same with various other protocols.

- *Fail-stop fault:* The fault that arises when a node halts its operation and alerts its neighbors of this fault [5].

- *Crash fault*: The fault that occurs if a node loses its internal state, or the node is damaged completely. A crash faulty node cannot participate in network activities, and it permanently stops responding to any triggering events. Fail-stop fault is a specific case of crash fault where computation halts, but not communication.

- *Omission fault*: The fault that arises when a node fails to send a required message on time or at all (*send omission fault*), or fails to receive a required message or behaves as the message had not arrived (*receive omission fault*). A node affected by omission faults stops responding to the triggering events, occasionally or permanently. Such faults occur mainly due to the faulty transceiver, a link that occasionally loses message, etc. Crash fault is a subclass of omission fault class.

- *Timing fault*: A fault that occurs when a node completes a task correctly, but either too early (probably because of a timer that runs too fast), or too late (probably due to excessive message transmission, or processing delay because of processor or network overhead), or never [6]. The task may be a computational task, or simply a task of sending or receiving a message. In case, it never completes the task of sending or receiving a message, then timing fault behaves as an omission fault. So omission fault is a subclass of timing fault class.

- *Incorrect computation fault*: If a node fails to generate correct computational results given the correct set of inputs, then it is said to be affected by incorrect computation faults. Even though a node produces correct result, but delivers out of the expected time interval (*i.e.*, timing fault), it is categorized into incorrect computation faults [7].

- *Byzantine fault*: The class of all possible faults in a system is said to be Byzantine fault class. Each of the above mentioned fault class is a subclass of this universal fault class.

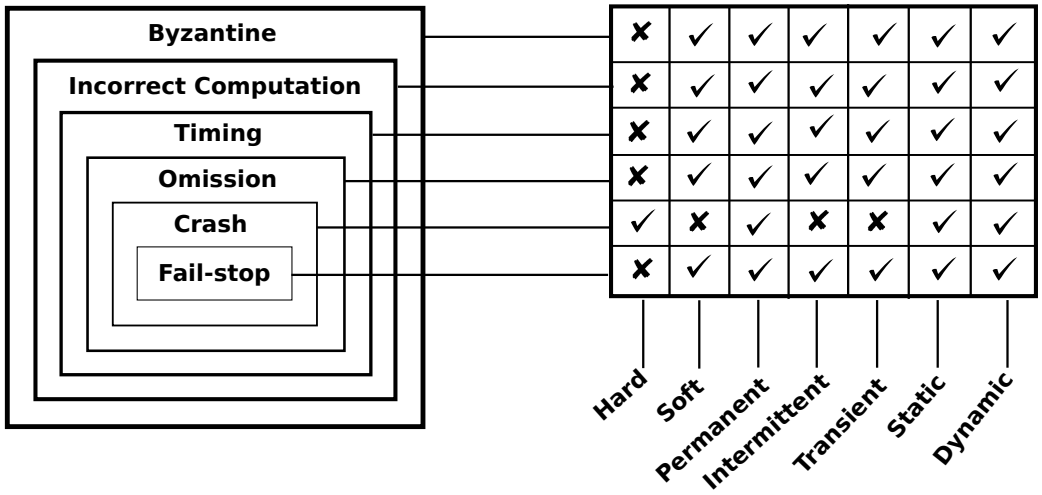


Figure 1.2: Fault Classification [3]

An ordered classification of above fault types is depicted in Figure 1.2. Crash faults are hard faults, and all others are put into the category of soft faults [8]. All faults can be permanent in nature; however, faults other than crash fault may appear transiently or intermittently. All these faults may appear before or during the diagnosis session.

1.2 Major Research Directions in Fault Diagnosis

Fault diagnosis is a broad area of discussion. It provides scope for further research in many ways. Following are the key problems:

(i) *Characterization problem:* Finding necessary and sufficient conditions in order to achieve the desired diagnosability in a system is the characterization problem. Here, *diagnosability* is the maximum number of faulty nodes that can be diagnosed unambiguously. This is also termed as *synthesis problem* [9].

(ii) *Diagnosability problem:* Determining the maximum number of faulty nodes (*i.e.*, diagnosibility) that the system can tolerate and identify unambiguously is the diagnosability problem.

(iii) *Diagnosis problem:* The diagnosis problem is to identify the correct set of faulty units in the system. It is important because incorrectly and/or inconsistently diagnosing the units in the system may degrade the Quality of Service (QoS).

(iv) *Diagnosis overhead problem:* The diagnosis overhead problem characterizes the message and time complexity along with energy consumption issues in diagnosing the system [8, 10, 11].

This thesis mainly concentrates on the *diagnosis problem*. The intrinsic characteristics and behavior of wireless networks makes the diagnosis difficult as compared to their wired counterparts. Therefore, we found it challenging to explore fault diagnosis in wireless networks.

1.3 Diagnosing Wireless Networks: Issues and Motivation

The diagnosis protocols designed for traditional wired networks cannot easily be propagated to wireless networks due to the following issues.

(i) They cannot take the advantage of the shared nature of communication in wireless networks since wired communication networks employ one-to-one communication paradigm.

(ii) Increase in link error probability in wireless networks, due to signal propagations affected by fluctuating environmental conditions, makes the network topology dynamic and unpredictable. Furthermore, node mobility aggravates this problem.

(iii) Wireless networks are usually deployed with limited resources such as battery power and bandwidth that puts a tight constraint on the amount of management traffic overhead the network can tolerate.

(iv) Wireless communications are vulnerable to link attacks [12]. Hence, malicious parties[1] can inject bogus or false information to the network, that may disrupt or interfere the diagnosis process.

[1]Malicious attacks and their mitigation mechanisms are outside the scope of this thesis.

1.3.1 Motivation

The above mentioned issues propelled us to investigate and devise new fault diagnosis protocols for wireless networks, ensuring smooth and continued operation. In particular, we have focussed on the widespread Mobile Ad hoc Networks (MANETs) and Wireless Sensor Networks (WSNs). Below given are some additional motivational points towards this research.

(i) Development of fault diagnosis protocols for wired interconnected networks are well studied over last four and half decades. However, fault diagnosis in wireless platforms has not taken a concrete shape yet, and is a matter of ongoing research.

(ii) Diagnosing and excluding faulty nodes from the network not only reduces traffic, but also avoids unnecessary energy consumption for relaying traffic from faulty nodes. Moreover, having the knowledge of faulty nodes in the network, further recovery actions can be taken. For example, a faulty node can be replaced by one of its replicas, if available. This in turn improves the network services like data delivery, availability, etc.

(iii) Intermittently faulty nodes, unlike permanently faulty ones, may behave correctly at some point and subsequently detected as fault-free. Therefore, the behavior of these faults needs to be studied vigilantly for correct diagnosis.

(iv) Transient faults are the manifestation of external errors that appear for very short span of time and their recurrence rate is very low. They are often treated fault-free. Isolating such fault-free nodes from the network affects the availability of resources and computation. Therefore, discrimination of transient faults from intermittent and permanent faults is highly important.

(v) Unlike MANETs, sensor networks are often deployed in unattended and hostile environments. This imposes a critical requirement to study fault diagnosis in WSNs.

1.4 System Model

MANET has grounded its importance in many applications ranging from small personal area networks to more critical military and rescue operations [13, 14]. Laterally, WSNs have received popularity over a broad spectrum of application such as industrial automation, environmental monitoring, space exploration, military surveillance, emergency health care, national security, habitat monitoring, etc. [15, 16, 17, 18]. However, instead of application specific models, we consider a generic system model of wireless network for our discussion.

1.4.1 Network model

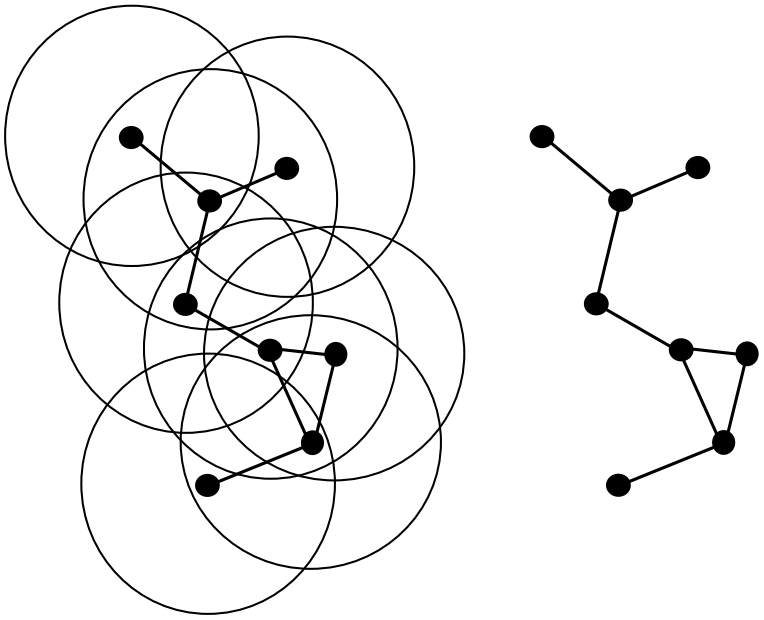
We consider a wireless network, consisting of a finite collection (say n number) of nodes. The nodes are assumed to be homogeneous, and the transmission range of all the nodes are the same. By homogeneous, we mean the nodes have the same initial energy, and similar storage and computing resources. Each node has a unique identity and they communicate via a multi-hop packet radio network. At the time of deployment the nodes are assumed to be healthy, *i.e.*, fault-free; but they may become faulty in due course of time. Each node has the knowledge of the identity of its 1-hop neighbors, at all times.

1.4.2 Communication model

The undirected graph $C(S, L^t)$, where S is the set of nodes and L^t denotes the set of logical links between them, represents the *communication graph* or *topology* (Figure 1.3) of the wireless network. We use the superscript t to quantify the attribute at a given time t; however, they may be removed from the notations when there is no explicit requirement of the notion of time.

Let $(S^t_{(i,x)}, S^t_{(i,y)})$ represent the Cartesian coordinates of the node S_i. Pair of nodes; S_i and S_j are said to be *adjacent* or *1-hop neighbors*, iff the Euclidean distance between them,

$$d^t_{(S_i,S_j)} = \sqrt{(S^t_{(i,x)} - S^t_{(j,x)})^2 + (S^t_{(i,y)} - S^t_{(j,y)})^2} \tag{1.1}$$



(a) (b)

Figure 1.3: An 8-node wireless network (left), and the corresponding communication graph (right)

does not exceed the communication range t_x, *i.e.*,

$$l^t_{(S_i,S_j)} \in L^t \Longleftrightarrow d^t_{(S_i,S_j)} \leq t_x. \tag{1.2}$$

Since the links in the communication graph are undirected, we have

$$l^t_{(S_i,S_j)} \in L^t \Longleftrightarrow l^t_{(S_j,S_i)} \in L^t. \tag{1.3}$$

The set of nodes adjacent to S_i at time t (denoted by $N^t_{S_i}$), called the neighborhood set of S_i can be defined as follows,

$$N^t_{S_i} = \{S_j | S_j \in S \text{ and } l^t_{(S_i,S_j)} \in L^t\}. \tag{1.4}$$

1.5 Performance measures

The proposed protocols are evaluated using two analytical measures: *Correctness*, and *Completeness*, adjoined by two quantitative measures: *Detection Accuracy (DA)*, and *False Alarm Rate (FAR)*. These metrics are defined as follows.

- **Correctness:** The fault diagnosis protocol is said to be *correct* if no faulty

node is diagnosed as fault-free nor vice-versa.

- **Completeness:** The fault diagnosis protocol is said to be *complete* if no node is left undiagnosed, and every node has the status of every other node in the network.

- **DA:** It is defined as the ratio of the sum of the number of faulty nodes detected as faulty and the number of fault-free nodes detected as fault-free to the total number of nodes in the network.

- **FAR:** It is the ratio of the sum of the number of fault-free nodes diagnosed as faulty and the number of faulty nodes diagnosed as fault-free to the total number of nodes in the network.

1.6 Contributions

In this thesis, diagnosis protocols for MANETs, and for more resource constrained and error prone WSNs are proposed. In particular, the contributions are as follows:

(i) Design and evaluation of online distributed fault diagnosis algorithm to handle hard and soft faults in MANETs. Faults are permanent in nature and the topology of the MANETs does not change during diagnosis.

(ii) Design and evaluation of online distributed fault diagnosis algorithm to handle permanent as well as intermittent faults in MANETs. Dynamic topology of MANETs is considered.

(iii) Design and evaluation of online distributed fault diagnosis algorithm to handle permanently, and intermittently faulty nodes in WSNs. Impact of threshold values on DA and FAR are discussed.

(iv) Design and evaluation of online distributed fault diagnosis algorithm to discriminate transient faults from intermittent and permanent faults. Instead of excluding transiently faulty nodes, they are to be reintegrated back to the network.

(v) Verification of the correctness and completeness of the proposed protocols with the help of mathematical analysis.

(vi) Validation of the proposed diagnosis protocols using INET-20111118 (for simulating MANET protocols) and Castalia-3.2 (for simulating WSN protocols) simulators, both based on the OMNeT++ 4.2 platform.

(vii) Evaluation of the efficiency of the proposed protocols using two mostly used performance metrics, *viz.*, DA and FAR.

1.7 Thesis Organization

The thesis is organized as follows.

Chapter 1: Introduction

This chapter explains various issues and need of fault diagnosis in wireless networks.

Chapter 2: Literature Review

This chapter brings together the pioneer works in the field of fault diagnosis, that are already in place. The existing protocols are studied and their loopholes are adverted.

Chapter 3: Permanent Fault Diagnosis in Static Topology MANETs

This chapter introduces an online distributed fault diagnosis protocol with an aim to detect permanently faulty nodes in MANETs, where nodes do not move out of the transmission range of its neighbors during diagnosis session. Both hard and soft faults are entertained.

Chapter 4: Intermittent Fault Diagnosis in Dynamic Topology MANETs

This chapter presents an online distributed fault diagnosis protocol to handle both permanent and intermittent faults in MANETs. Furthermore, the nodes are allowed to move out of the transmission range of its neighbors during diagnosis session.

Chapter 5: Intermittent Fault Diagnosis in Static Topology WSNs

This chapter introduces an online distributed fault diagnosis protocol to detect and remove intermittently faulty sensor nodes from WSNs. The topology is static throughout the diagnosis session. In addition, the impact of values of thresholds on the performance is studied.

Chapter 6: Adaptive Fault Characterization Based on their Persistence in Static Topology WSNs

With the observation that transiently faulty nodes are affected by external faults and often considered fault-free, this chapter presents an adaptive fault characterization protocol based on count-and-threshold mechanism to discriminate transient faults from intermittent, and permanent faults.

Chapter 7: Conclusions and Future Work

This chapter provides the concluding remarks of the work. The scopes for further research are outlined at the end.

Till now we have seen the bird's eye view of fault diagnosis in wireless networks and the contributions in this thesis. These contributions are discussed more concretely, in subsequent chapters, in sequel. Each chapter highlights the proposed protocol, the analytical and simulation results, and comparative analysis.

Chapter 2

Related Works

The *as-is* analysis helps in understanding the current state-of-the-art approaches, followed by the *to-be* analysis that articulates further requirements or enhancements. This chapter presents the *as-is* analysis of the existing diagnosis mechanisms for wireless networks, with an introduction to some elementary approaches in wired platforms. The shortcomings of the extant protocols are analyzed, and marked for *to-be* augmentation.

The diagnosis protocols in the literature can be broadly classified into two main categories, based on the number of steps followed to diagnose the whole system:

(i) *One-step diagnosis:* These diagnosis protocols aim to identify all faulty units in a system at a time, *i.e.*, in one step, with a constraint on the maximum number of allowed faults.

(ii) *k-step diagnosis:* These diagnosis protocols try to identify one faulty unit, and after it is repaired or replaced, the testing continues for k steps to eventually identify all faulty units, provided the number of faulty units does not exceed a maximum value, say t.

Depending on whether the diagnosis is carried out by a single or all nodes in the network, the diagnosis protocols can be any one of the following two categories:

(i) *Centralized diagnosis:* These diagnosis protocols employ, geographically or logically, an ultra-reliable node as central supervisory that takes the responsibility of fault diagnosis of the whole system. The central node collects the responses from

the nodes of the network after sending them periodic diagnostic test messages. The nodes are classified as faulty or fault-free after analyzing these responses.

(ii) *Distributed diagnosis:* In this type of diagnosis protocols, every node in the network participates in the diagnosis process. There is no central observer. Each node tests a group of other nodes, usually the 1-hop neighbors, to generate a partial diagnosis view. These partial views are exchanged to generate the global and complete view.

Conditional upon to perform diagnosis during the normal system operation, or when the system is idle, fault diagnosis protocols are of two types:

(i) *Online diagnosis:* Fault diagnosis is said to be online, if the diagnosis is carried when the system is operating under working conditions at its service area.

(ii) *Offline diagnosis:* Offline diagnosis is carried out when the system is not operating, *i.e.*, during maintenance hours.

A taxonomy of fault diagnosis protocols is shown in Figure 2.1. The butterfly connections in the figure conveys that a diagnosis protocol can be any of the eight possible combinations of these categories.

Centralized diagnosis protocols may be efficient for small networks; however, they are non-scalable and cannot be advantageous for larger networks. In such protocols, the diagnosis latency is expected to be more. In addition, the central observer may be the bottleneck due to high traffic, especially in case of energy constraint networks. Therefore, more emphasis is given on auditing generic distributed diagnosis protocols.

2.1 System Level Diagnosis−the early approaches

System level diagnosis brought a radical change in the field of fault tolerant computing with the objective of identifying unit's state, *i.e.*, faulty or fault-free. In order to reach this objective one or more tests are applied to the units in the system. The collection of the results of these tests is called a *syndrome*. The syndrome is then analyzed to deduce the correct and consistent fault set. Once the faulty units have been identified, the system is able to isolate them and ignore their outputs, if any.

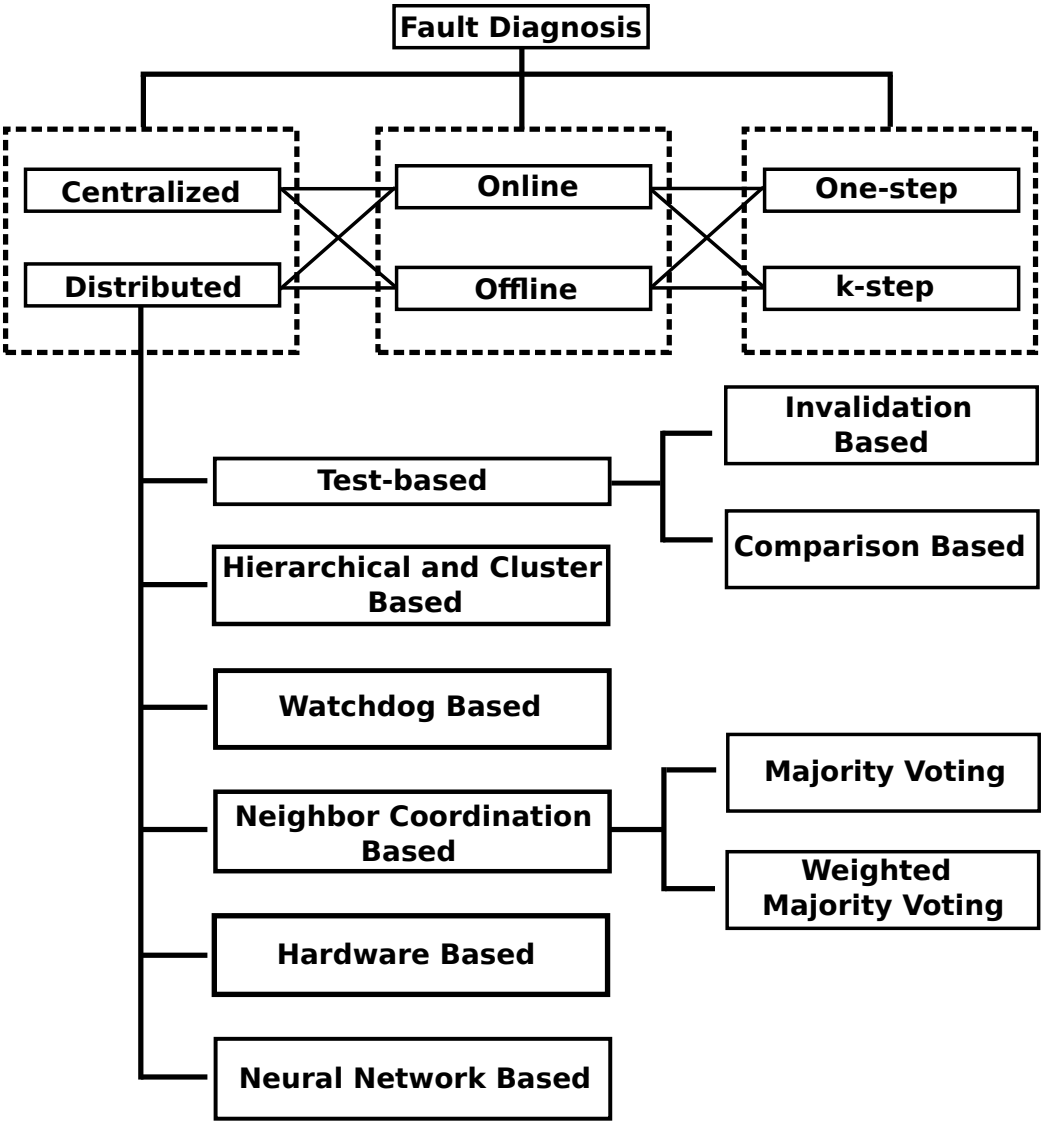


Figure 2.1: Taxonomy of fault diagnosis protocols.

The light into the area of system level fault diagnosis was first put by Preparata, Metze, and Chien in 1967 [19], for a wired network with point to point communication links. Thenceforth, the model is referred to as PMC model, after the names of the authors. In this model, the units are required to perform mutual tests exploiting systems interconnection. The unit being tested, say S_j, generates result of the test task received from its tester, say S_i. Once S_i receives the result from S_j, it can match this with the expected result to decide if S_j is faulty (match outcome is 1) or fault-free (match outcome is 0). Of course, the test is reliable only if the tester is fault-free. This model is also known as *symmetric invalidation model* [20, 21]. Thereafter, the PMC

model has been used by many researchers as a basis to devise several fault diagnosis protocols [3, 9, 20, 21, 22, 23, 24]. A similar test-based model has been proposed by Barsi, Grandoni, and Maestrini in 1976, called BGM model [9]. However, this model differs from the PMC model with respect to fact that, in the BGM model if the tester node and the node being tested, both are faulty then the test outcome is always 1; whereas in case of PMC model it is unpredictable. This model is also known as *asymmetric invalidation model* [20, 21].

One of the most propitious approaches in the direction of test-based system level diagnosis is *comparison approach* which has been proven to be efficient against performing mutual tests. The first comparison based diagnosis model was proposed by Miroslaw Malek [25], where the output of the same task, generated by pair of units, say S_i and S_j, are compared. The comparison outcome $c_{(S_i,S_j)}$, with respect to a reliable comparator can be put into Equation 2.1.

$$c_{(S_i,S_j)} = \begin{cases} 0, & \text{if both units } u \text{ and } v \text{ are fault-free} \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

This is also called as *asymmetric comparison model*, based on the assumption that no faulty unit generates same wrong result as any other faulty unit in the system. In contrast, *symmetric comparison model* proposed by Chwa and Hakimi [26] assumes that two faulty units may generate the same result, making the comparison outcome unpredictable in such cases.

Table 2.1: Testing rules for Test-based models

		Invalidation models		**Comparison Based models**	
S_i	S_j	Asymmetric	Symmetric	Asymmetric	Symmetric
Fault-Free	Fault-Free	0	0	0	0
Fault-Free	Faulty	1	1	1	1
Faulty	Fault-Free	X	X	1	1
Faulty	Faulty	1	X	1	X

The testing rules for invalidation and comparison based models are shown in Table 2.1. Symbol "X" in the table indicates an unpredictable outcome. The comparison

based approach is further amended in the literature [27, 28, 29, 30, 31]. Variants of these approaches are applied in wireless networks to diagnose faults.

2.2 Distributed Diagnosis Protocols for Wireless Networks

Demand for online fault diagnosis with low message overhead, and high accuracy; in conjunction with the proficiency of distributed computing shifted the paradigm of fault diagnosis from centralized to distributed. By mitigating the limitations of centralized approach, the distributed protocols made themselves easily scalable to larger and denser networks. This section brings together the existing distributed system level fault diagnosis protocols for MANETs, and WSNs.

2.2.1 Test-based Approaches

In the literature, many researchers have focused on the formulation of test-based techniques to handle faults in wireless networks. Capturing and examining the responses of the test messages form the basis of such diagnosis techniques.

Chessa and Santi were the first to develop Distributed Self Diagnosis Protocol (DSDP) for ad hoc networks [10]. Their diagnosis protocol, called *static DSDP*, assumes a constraint on node's mobility. They cannot migrate outside the transmission range of their neighbors during the diagnosis session, *i.e.*, the network topology remains static. In order to test its neighbors at time t, a fault-free mobile S_i sends them a test message, *i.e.*, $msg = (S_i, k, T_k)$, where k is the sequence number of the test task T_k. Upon receiving the test message any node $S_j \in N^t_{S_i}$ that is not hard faulty sends back a response message carrying the result of the task T_k, at time t' such that $t < t' \leq t + t_{out}$. The response message is of the form $msg = (S_i, k, R^k_{S_j})$, and t_{out} is the time-out period chosen such that the slowest neighbor is guaranteed to respond to the test request before t_{out} expires. Based on those responses a node can be diagnosed as faulty or fault-free using the asymmetric comparison based rules [9]. The local views at fault-free nodes are then exchanged, using a flooding based approach, to have global view about the fault status of the whole network at each

node. This model is able to diagnose the nodes at a cost of high communication complexity resulted from the requirement that a node responds to each incoming test request, and due to the adoption of flooding based dissemination process.

On the top of Chessa and Santi's model, three revised diagnosis protocols for MANETs have been asserted by Elhadef *et al.*, *viz.*, *dynamic DSDP*, *adaptive DSDP*, and *mobile DSDP* [8, 32]. The testing models for dynamic and adaptive DSDPs are similar to that of static DSDP. However, they differ with respect to the strategy followed during dissemination. Unlike static, dynamic, and adaptive DSDPs, mobile-DSDP considers time-varying topology of the network by unleashing the constraint on node mobility during diagnosis session.

In dynamic DSDP [8], a spanning tree is dynamically constructed covering all fault-free hosts in MANET, after testing phase. Dissemination of the local diagnostics start from the leaf nodes and propagates up in the tree to the root node. The root, after receiving all the local views from the fault-free mobiles, generates the global view and disseminates the same to all nodes down the tree. In adaptive DSDP [32], an initially configured spanning tree that covers all hosts in the MANET is maintained connected throughout the diagnosis session. Once the testing phase is done, a node removes all faulty nodes from its children. If its parent is found to be faulty then it sends a reconnect message to all its neighbors seeking a new fault-free parent. The spanning tree based dissemination reduces the communication complexity as compared to the flooding based approach used in static DSDP. In both dynamic and adaptive DSDPs, mobiles are required to respond to $(\sigma + 1)$ test messages, where σ is the connectivity of the network.

The authors in [32], have also discussed a mobile DSDP considering the time-varying topology of the MANET. A mobile host S_j, after receiving a test message from its neighbor S_i, sends the response including the test task. The rationale is that even if S_j moves out of the communication range of its tester S_i, any other host, say S_k, in its new neighborhood will be able to diagnose its status by executing the test task and comparing the result with the one included in the response. It also uses the flooding based dissemination strategy, similar to static DSDP, to disseminate the

diagnostic messages.

In all four above discussed protocols, the authors have mainly focused on detecting permanently hard and soft faults in MANETs, but intermittent and transient faults are not entertained.

A distributed fault diagnosis protocol called WSNDiag has been proposed by Chessa and Santi [11] to diagnose hard (crash) faulty nodes in WSNs. In this protocol, diagnosis process is initiated by a unique fault-free sensor node, upon receiving an explicit request from an external operator. During the diagnosis period, two types of messages are exchanged: *I'm alive* (IMA) messages, and diagnostic messages. A spanning tree that covers all fault-free sensor nodes in the network is constructed dynamically, during the dissemination of IMA messages. The sensor nodes that did not reply with their IMA messages within a predefined time-out period t_{out}, are diagnosed as faulty. After t_{out} expires, a node S_i has the local view constituting the fault status of its neighbors. Sensor node S_i collects the local views of its children, and transmits the combined view to its parent. Once the initiator receives the local diagnosis views from all of its children, it generates the global view by aggregating these local views with its own. The global view is then transmitted down the tree to all fault-free sensor nodes, to reach a general consensus.

Weber *et al.* have adopted the invalidation model to propose a testing strategy among the nodes in a WSN, in order to ensure that the desired diagnosability of the system is met [33]. Based on whether reciprocal tests are allowed or not, they have devised two heuristics. In case reciprocal tests are not allowed, the region of interest is divided into four equal sized quadrants and each node S_j in a particular quadrant is tested by at least $(t+1)$ other nodes in its preceding quadrant ensuring t-diagnosability. In case reciprocal tests are allowed, the degree of the network must be greater than t to ensure t-diagnosability. This is achieved by constructing the diagnostic graph with all possible tests among the sensors.

Weber *et al.* have extended their previous work [33], and proposed an energy efficient test connection assignment model for WSNs [34]. Unlike their previous approach, in this strategy, sensors execute predefined tasks and send the outputs to their

Table 2.2: Test-based Diagnosis Protocols

Protocol	Year	Key Points	Remarks
Static DSDP, Chessa & Santi [10]	2001	• First DSDP for ad hoc networks. • Nodes respond to all incoming test messages. • Employs flooding based dissemination.	• Considers static topology only. • High communication complexity. • Does not consider temporary faults.
Adaptive DSDP, Elhadef *et al.* [32]	2006	• Introduces a spanning tree based dissemination strategy. • Nodes respond to $(\sigma+1)$ tests. σ: connectivity. • Low communication complexity compared to static-DSDP.	• Topology does not change during diagnosis. • Detection accuracy reduces compared to static-DSDP. • Concentrates on permanent faults only.
Mobile DSDP, Elhadef *et al.* [32]	2006	• Time-varying topology is focused. • Nodes respond to $(\sigma+1)$ tests. • Response message also includes the test task.	• Results in reduced DA. • Temporary faults are overlooked.
Dynamic DSDP, Elhadef *et al.* [8]	2008	• Spanning tree is constructed during diagnosis. • Nodes respond to $(\sigma+1)$ tests. • Less maintenance cost for the spanning tree compared to adaptive-DSDP.	• Does not consider dynamic topology. • Handling temporary faults are not discussed.
WSNDiag, Chessa and Santi [11]	2002	• Responding to *I'm Alive* message is the basis of diagnosis. • Spanning tree based dissemination is used. • No extra communication cost for tree construction since *I'm Alive* message does the job.	• Static topology is assumed. • Only hard faults are diagnosed.
Weber *et al.* [33]	2010	• Problem of determining proper test strategy is discussed to ensure t-diagnosability. • With and without reciprocal test scenarios are considered.	• The level of diagnosability is dependent on the selected testing edges. • Diagnosability reduces by restricting the tester of a node from a particular quadrant.
Weber *et al.* [34]	2012	• Sensors execute a predefined diagnosis task and sends the result to its testers. • With the knowledge of geographical locations of sensors, distance between tester and testing nodes are minimized.	• Mutual reciprocal tests are not considered. • Moreover, the diagnosability reduces by restricting the tester of a node from a particular quadrant.

testers in the preceding quadrants. The geographical locations of the sensors are also taken into account to minimize the distance between tester and tested sensors, ensuring tests with lesser energy cost. The test-based diagnosis protocols are summarized in Table 2.2.

2.2.2 Hierarchical and Cluster Based Approaches

The basic idea of hierarchical diagnosis mechanisms is to maintain all the nodes in the network in a layered fashion. Most of the protocols adapt spanning tree based hierarchical test graph modelling. The tests are performed by the parents on the children. The diagnostics are disseminated along the edges of the spanning tree.

A cluster based protocol divides the whole network into a number of groups, called clusters. Each cluster is monitored and controlled by a Cluster Head (CH). Each CH executes the fault detection algorithm on its members.

In order to alleviate the *response implosion problem* in case of a centralized diagnosis scheme, Jaikaeo *et al.* have proposed a cluster based distributed approach, called *diffused computation* [35], for WSNs. The initiator triggers the diagnosis process at the CHs. The CHs then pool the sensor readings from their member nodes. The difference between each reading and the average reading for all the members are compared with a predefined threshold. If the difference exceeds the threshold, then the sensor is identified as faulty. After diagnosis, the CHs send their local diagnosis information to the manager.

Gupta and Younis [36] have focussed on the detection of faulty gateways in clustered WSNs. A gateway node, after receiving the sensor data and energy values of all its members, generates "*Status*" message encompassing the information of all its in-house sensors and the status about the gateway itself. These status messages are exchanged periodically between the gateways, exploiting inter-gateway communication. A gateway is not considered completely failed, if at least one of the gateways in the network is able to communicate with it. Following this, communication faults are tolerated. Considering the status messages from the gateways as their heartbeat messages, gateway faults are handled. Sensors in the vicinity of a faulty gateway are recovered by peering them to another cluster, based on the backup information kept

during the clustering phase.

Cluster based failure detection mechanism for ad hoc wireless networks is the key discussion in [37]. The authors have presented a heartbeat style detection mechanism. The members of the cluster along with the CH generate periodic heartbeat messages. The heartbeats of a member node can be heard by all its 1-hop neighbors including the CH. A member, after hearing the heartbeats of its neighbors, notes them in a digest message and sends it to the CH. A host S_j is detected faulty, if the CH receives neither heartbeat nor the digest message from it, and none of the digests that the CH receives reflect a members awareness of the heartbeat of S_j. The CH then broadcasts a health status message embedding the fault status of the member hosts. The health of the CHs are monitored by the spare CHs called Deputy Cluster Heads (DCHs). A DCH detects a CH as fault-free, if (i) it hears the heartbeat message, digest message, or health status message from it, or (ii) any other node has intimated about the liveness of this CH through a digest message.

Iskander *et al.* [38] have suggested a proactive management architecture (PMA) to predict failure, and move the tasks from such nodes to the safe nodes. The PMA is composed of a supervisory node and a set of agencies. Each agency is a group of nodes with a main manager which acts as the cluster head. The architecture is used to predict the future state of the nodes. Thus, faulty nodes can be detected by comparing the current state with the predicted state.

The networks where members of a cluster go through duty cycle may not hear periodic CH heartbeats which helps in detecting the liveness of the CH. This issue has been resolved by Ossama *et al.* [39]. In this case, a regular node can solicit a heartbeat from its CH after sending a certain number of messages.

Rost and Balakrishnan have proposed a detection algorithm called Memento [40]. In their hierarchical approach, a node is required to monitor its children in the topology. Memento performs failure detection using heartbeat interface and a failure detector at each node. Each parent listens to the periodic heartbeats from its children. The failure detector decides the liveness of nodes it monitors, and records it in a liveness bitmap. The failure detector at each node basically checks the loss rates of

Table 2.3: Hierarchical and Cluster Based Diagnosis Protocols - I

Protocol	Year	Key Points	Remarks
Diffused Computation, Jaikaeo *et al.* [35]	2001	• Manager node triggers the diagnosis process at each CHs. • Difference between each sensor reading and the average of all member sensor readings are compared with a threshold to diagnose faulty nodes.	• Diagnosis of CHs are not emphasized. • Presence of faulty CHs may drastically reduce the accuracy.
Gupta and Younis [36]	2003	• Concentrates on the detection of faulty gateways in clustered WSNs. • sensors in the vicinity of a faulty gateway are peered with another healthy gateway.	• Mobility of gateways are not emphasized. • Usefulness of sensors transmitting their energy level to the gateways are not shown.
Tai *et al.* [37]	2004	• Diagnosis is done at the cluster level, and complete system view is obtained through inter cluster diagnostic exchanges. • To reduce the false alarms, in case heartbeat message loss, each node is required to hear the heartbeats of its neighbors and notify the CH about it.	• Simulation results for large networks and the effect on DA are not observed in the discussion. • Neither intermittent and transient faults nor the time-varying topology are dealt.
PMA, Iskander *et al.* [38]	2004	• The architecture is composed of a supervisory node and a set of main managers (CHs). • Faulty nodes are detected by comparing the current state with the predicted state.	• Diagnosis of main managers are not addressed.
Ossama *et al.* [39]	2005	• The contention that a regular node in sleep mode cannot hear the periodic CH heartbeats to decide its fault status, is resolved. • A member, after sending a certain number of messages can solicit a heartbeat from its CH.	• Concentrates on end to end delivery rather than on general consensus. • Deals with CH faults only.
Memento, Rost *et al.* [40]	2006	• Heartbeat based detector. • A parent, if observes the heartbeat loss rate of a child goes beyond a threshold, detects the child to be faulty.	• Considers fail-stop faults only. • Not properly tuning the allowed heartbeat loss rate may raise false alarms with reduced DA.
CBCD, Dongni Li [41]	2007	• CHs are diagnosed first following comparison based diagnosis model [10]. • Nodes within the clusters are diagnosed by respective CHs using heartbeat message exchanges.	• Behavior with respect to large networks are not discussed. • CBCD with dynamic network topology has not been simulated.

heartbeats from its children. If this rate for any child, say S_j, exceeds a predefined threshold, then parent, say S_i, detects that child as faulty. At this point, S_i updates the j^{th} bit of its liveness bitmap to 1. The parent performs bitwise OR operation (*i.e.*, aggregation) on the local liveness bitmaps of its children together with its own bitmap and forwards it up in the tree until it reaches the initiator.

Dongni Li has proposed Cluster Based Comparison Diagnosis (CBCD) model [41] for hierarchical ad hoc networks. The CHs are diagnosed by applying the comparison based diagnosis model of Chessa and Santi [10], on a logical subnet formed by picking out all the CHs and the paths connecting them as virtual links. The CHs take the obligation to diagnose the mobile hosts within the clusters. The diagnosis views generated at each CHs are then exchanged among themselves to have a complete view of the system's state.

Diagnosis in underwater sensor networks has been investigated by Wang *et al.* [42]. The authors have proposed an agreement-based mechanism for detecting faulty CHs. Each cluster member maintains a status vector in which each bit is initialized to zero. Each bit in the status vector corresponds to the decision of a member node about the fault status of the CH. If a member does not hear the heartbeat from the CH for a specific period of time, then it detects the CH to be faulty and sets the corresponding bit in its status vector to 1. Individual status vectors are exchanged following a spanning tree based mechanism to reach an agreement. If the status vector of a member does not contain a zero bit, then the CH is considered failed.

Detecting sensor nodes suffering from energy exhaustion has been studied by Venkataraman *et al.* [43]. The nodes in each cluster are required to convey their current energy status through a $hello\_msg$ to all their 1-hop neighbors. When the energy level of a node falls below a threshold value, it sends failure report message to its parent and children. The threshold value is the energy required to transmit D number of messages across a distance t_x, where t_x is the transmission range and D is the maximum number of 1-hop neighbors selected during clustering. This may make some nodes to be disconnected from the cluster. The parent and children of the failing node take necessary action to reconnect these nodes back to the cluster.

Table 2.4: Hierarchical and Cluster Based Diagnosis Protocols - II

Protocol	Year	Key Points	Remarks
Wang *et al.* [42]	2007	• Before deciding the CH's status, each member node consults with all other members of that cluster. • Copes with the loss of heartbeat messages from the CHs.	• The members must be synchronized with their CHs to be able to wakeup to capture heartbeats. • Deals with CH faults only.
Venkataraman *et al.* [43]	2008	• A node, if finds its remaining energy falls below a threshold, sends a failure report message. • The parent and children of the failed node take necessary action for connectivity recovery.	• Deriving a general fault consensus is deferred. • Connectivity recovery in case of failure is the main goal. • Fault due to energy exhaustion is considered.
Asim *et al.* [44]	2008	• If a cell manager does not receive any update from a member node, then it declares that node to be faulty. • If a group manager does not receive the health update from a cell manager, the cell manager is considered to be faulty.	• Considers permanent faults (hard type only). • Very similar to other cluster based approaches. • Results are not validated through simulation.
Li *et al.* [45]	2010	• CH failures are handled by a backup CHs. • Intra cluster diagnosis is similar to CBCD [41]. • Inter arrival time of heartbeat messages are predicted dynamically to reduce the false alarms.	• The impact of dynamically predicting and updating the heartbeat message inter arrival time has not been shown through simulation. • Behavior with respect to large and dynamic networks are not discussed.
ATMP, Gheorghe *et al.* [46]	2010	• Employs spanning tree based testing and dissemination. • Sensor measurement is said erroneous if it deviates more from the median of the measurements in that cluster.	• Considers only data faults. • Behavior with respect to changing topology is left out of the discussion.
Khan *et al.* [47]	2010	• Network is divided into four zones, each with a zone manager and many clusters. • Zone managers are to diagnose CHs, which are further responsible to diagnose the members.	• Node mobility is not considered. • No supportive simulation results.
Taleb *et al.* [48]	2010	• Tests are performed in the test trees following in-order traversal. • A child node is considered faulty if it does not provide the expected result.	• Maximum one fault per cluster; which may not be practical since in an event region more number of neighboring nodes may be faulty.

A cellular-based approach for permanent (hard) fault detection in WSNs has been proposed by Asim *et al.* [44]. The network is divided into grids, called cells. Each cell is treated as a cluster, and has a cell manager and a gateway node. The gateway node along with the sensors within a cell send their updates that include the node ID and the remaining energy level, when they are asked by the cell manager. If the cell manager does not observe any such updates from a node, then it sends an instant message to that node and acquires its status. In return, if the cell manager does not receive the acknowledgement in a specific time bound, it declares the sensor node to be faulty and conveys this to all other nodes in the network. Each cell manager transmits its health status information to its group manager through the gateway node. If the same is not received by the group manager then the cell manager is considered faulty.

A fault detection service for clustered ad hoc networks has been put by Li *et al.* [45]. The working of this approach is similar to the CBCD model discussed by Dongni Li [41]. The difference is that, this approach does not consider the diagnosis of CHs. Instead, they considered a backup CH that overtakes the job of the main CH in case of failure. The intra cluster diagnosis is performed using heartbeat message exchange mechanism. Due to change in network status, the inter arrival duration between heartbeat messages may change. For this, the authors have formulated a heuristic to dynamically predict this inter arrival time, thus avoiding the erroneous decision of fault detection upto some extent.

The Adaptive Trust Management Protocol (ATMP), proposed by Gheorghe *et al.* [46], works in three phases: setup phase, learning phase, and the exchange phase. In the setup phase, a spanning tree is constructed that overlay all nodes in the network. In the learning phase, fault detection is carried out. Every node stores the data received from its children. The children are then grouped into clusters such that the distance between nodes within a cluster is less than a predefined constant. The measurements of the nodes within a cluster are put into a list in non-decreasing order. The measurement of a node in the list is considered erroneous if its deviation from the median of that list exceeds a predefined constant. Based on this, the penalty and

reputation values are modified. Reputation is the expectation about an individual's behavior based on information about, or observations of its past behavior [49]. In the exchange phase, the penalty associations and reputation values are disseminated that helps a parent node to determine a binary trust value.

Khan *et al.* [47] have presented a zone-based fault tolerant management architecture (ZFTMA) for WSNs. The network is divided into four symmetric zones, each associated with a zone manager. The nodes in each zone are grouped into different clusters. The CHs in each zone are monitored by their zone manager. The intra cluster members are diagnosed by their respective CHs, and the zone manager is responsible to diagnose the CHs.

Taleb *et al.* [48] have discussed a cluster based diagnosis protocol for sensor networks, considering no more than one fault per cluster. For each cluster, two test trees are generated. The testing is performed by traversing the trees in in-order fashion. A child node is considered faulty, if the result from it does not match the expected result.

Huang *et al.* [50] have considered the problem of dynamically choosing the probe sensors (the nodes that report to the CHs after performing fault diagnosis). Their approach exploits the *Pareto principle* that most of the faults are contained in a small number of clusters. Assigning more number of probe stations to such clusters increases the detection accuracy.

Wang *et al.* have presented a Cluster-based Real-time Fault Diagnosis Aggregation (CRFDA) algorithm [51] for WSNs. It is developed on the top of Chessa and Santi's model [10]. In this case, the tasks are assigned to the cluster members by the respective CHs. The task results are compared by the CHs to decide the fault status of the members.

Kazi Sakib has proposed an Asynchronous Failed Sensor node Detection (AFSD) mechanism [52] for clustered WSNs. Absence of the global clock makes the diagnosis more challenging. Each node maintains a numeric failure-counter variable for each of its neighbors to track the messages sent to and received from it. On the events of transmitting or receiving messages, AFSD modifies the corresponding failure-counters.

Table 2.5: Hierarchical and Cluster Based Diagnosis Protocols - III

Protocol	Year	Key Points	Remarks
Huang *et al.* [50]	2011	• Dynamically chooses the probe stations in clusters to improve the network lifetime. • Assigns more number of probe stations in clusters containing more number of faulty nodes.	• Neither the faults in CHs not the dynamic environment is considered.
Wang *et al.* [51]	2011	• Follows Chessa and Santi's model [10]. • CHs are responsible to assign the tasks to the member nodes, and compare their results.	• CH faults are not entertained. • Static topology is assumed during diagnosis.
AFSD, Kazi Sakib [52]	2011	• Compatible for asynchronous systems. • A failure-counter tracks the transmission and reception of messages. It increases beyond a threshold value for a failed neighbor.	• Value faults and temporary faults are not considered. • Dynamic topology of the network is avoided.

For a fault-free node, the value of the counter is bound and tends to zero, and for a faulty sensor node it is unbound and tends to infinity. If the value of this counter for a particular neighbor exceeds a predefined threshold, then the neighbor is suspected to be faulty. A consensus among the CHs is followed before declaring the suspicious nodes to be faulty. The hierarchical and cluster based algorithms for fault diagnosis are briefed in Table 2.3, 2.4, and 2.5.

2.2.3 Watchdog Based Approaches

In these approaches, each node overhears the transmissions of its 1-hop neighbors to check if they forward the packets sent to them or not. If a neighbor does not forward the packet, then it is suspected to be malicious. However, the node can be declared as faulty only when the misbehaving rate exceeds a predefined threshold.

Marti *et al.* [53] have presented a watchdog based approach to detect malicious nodes in MANETs. Each node maintains a list of recently sent packets and compare each forwarded packet with the packet in the buffer for a match. If a match is found, the packet is removed from the list. If a packet has remained in the buffer for longer

than a specific time-out period, the watchdog increments the penalty for that node. If the penalty for a node increases beyond a threshold, then the node is considered misbehaving.

Table 2.6: Watchdog Based Diagnosis Protocols

Protocol	Year	Key Points	Remarks
Marti *et al.* [53]	2000	• If a neighboring node does not forward the packets, it is considered misbehaving. • If the misbehaving rate exceeds certain threshold, it is considered malicious.	• If there is ambiguous collision at a particular node then it cannot overhear the transmissions from its neighbors. • An intruder may falsely report other nodes as misbehaving.
Patcha and Mishra [54]	2003	• Extension to [53], in order to handle nodes falsely reporting other nodes of being malicious. • Each watchdog, that is selected from set of trusted nodes, maintains two thresholds to decide a neighbor to be malicious or trusted.	• Keeping the SUSPECT_THRESHOLD at a reasonably high level may delay the detection of many trusted nodes. • Handling ambiguous collision problem is not considered.
Nasser and Chen [55]	2007	• A node after receiving report about any malicious node, inquires it from the destination. • The node accepts the report, if the destination confirms it; otherwise the reporting node is considered malicious.	• If the real malicious node is on all paths from source to destination, then the scheme fails. • Still cannot avoid ambiguous collision problem.

To handle the intruders that falsely report other nodes as misbehaving, Patcha and Mishra [54] have extended the concept of watchdog. In their approach, the first few nodes during network formation are assumed to be trusted nodes. Every normal node that joins afterwards has to prove its trustworthiness. The watchdogs are selected from the trusted nodes only. Each watchdog declares the neighboring nodes as malicious or trusted using two thresholds, namely, SUSPECT_THRESHOLD (a measure of misbehavior of the neighbors) and ACCEPTANCE_THRESHOLD (a measure of good behavior of the neighbors). To handle the same problem, Nasser and Chen [55] have extended the work of Marti *et al.* [53]. They have discussed an enhanced watchdog (ExWatchdog) mechanism. When a node (say S_j) intimates the source that its next hop (say S_k) is misbehaving, the source inquires it from the destination by sending a message using an alternative path. If the destination

confirms about the same then S_k is detected malicious; otherwise, S_j is considered good. However, if the malicious node S_j is on all paths from source to destination then the scheme fails. Table 2.6 gives a summary of watch-dog based fault diagnosis protocols.

2.2.4 Neighbor Coordination Based Approaches

Neighbor coordination has been extensively used by many researchers for designing fault tolerance algorithms for WSNs. In such approaches, nodes coordinate with their neighbors by exchanging their current status, such as energy level, sensor measurement, etc. All the neighbors of a node participate in deciding the status of that node, before exposing the faulty nodes to the base station. This results in less communication with the base station and hence, reduces the energy consumption. These approaches basically follow some majority voting protocols exploiting the fact that the faulty measurements are uncorrelated while the normal measurements are spatially correlated [56].

Krishnamachari and Iyengar [57] have proposed a Bayesian algorithm for recognizing sensor measurement faults. They have discussed a randomized decision scheme, a threshold decision scheme, and an optimal threshold scheme. Each node S_i finds an estimation of the true reading given its own sensor reading, along with the information that k neighboring sensors report similar reading as S_i. This estimation forms the basis for determining the measurement faults. This work has been further extended by Luo *et al.* [58] to consider sensor faults along with the measurement faults. For a given detection error bound, minimum neighbors are selected to minimize the communication overhead.

Ding *et al.* [59] have discussed a localized algorithm to identify faulty sensors. The deviation of the measurement of a node S_i is compared with the median of the measurements of the nodes within a certain radius from S_i. Such deviations are found for all nodes within a particular area. The nodes that are showing extreme deviations are treated as faulty.

Chen *et al.* [60] have proposed a distributed majority voting algorithm to identify faulty sensors. Each sensor S_i tests every neighbor S_j and generates a binary result

r_{ij}. If measurement variation between S_i and S_j at time t, *i.e.*, v_{ij}^t exceeds δ_1 then S_i monitors the change in v_{ij}^t from time t to $t+\Delta t$. If this change $\Delta v_{ij}^{\Delta t}$, exceeds δ_2 then r_{ij} is set to 1. If less than half of the neighbors set r_{ji} to 1 then S_i is considered likely good (LG); otherwise likely faulty (LF). Here, δ_1 and δ_2 are predefined thresholds. Considering $S_j \in N_{S_i}^t$ that are LG, if the number of such S_j bearing r_{ji} value 0 is more than the number of such S_j bearing r_{ji} value 1 by at least $\lceil |N_{S_i}^t|/2 \rceil$ then S_i is detected to be good or fault-free. All the undetermined sensors repeatedly check for $\sqrt{n}$ times (in average case), if any of its neighbors is fault-free. If such a neighbor exists, the test result of this neighbor is used to detect the actual fault status of the undetermined node. If still ambiguity occurs, the sensor's partial status is used to derive its actual status. If $\{S_j, S_k\} \in N_{S_i}^t$ are determined as fault-free and $r_{ji} \neq r_{ki}$ then S_i is detected as fault-free if its partial status is LG; faulty if it is LF.

Hsin and Liu in [61] have suggested a two-phase timer scheme for distributed self-monitoring in WSNs. In the first phase, if a node finds some of its neighbors not responding within a specific time bound, then in the second phase, it consults and coordinates with other neighbors to take a more accurate decision about the fault status of those neighbors. In this approach, active monitoring is used only between the neighbors, and network-wide passive monitoring is used so that control center is not made aware unless something is wrong.

In the distributed fault detection algorithm proposed by Lee and Choi [62], time redundancy is used to tolerate transient faults. Each node S_i tests every neighbor S_j for q rounds. Every node S_i maintains a result matrix R with q columns and rows for each of its neighbors. R_{jk} is set to 0 if $|x_i^k - x_j^k| < \delta_1$; otherwise it is set to 1. Here, x_i^k and x_j^k are the sensor readings of S_i and S_j in k^{th} round, and δ_1 is a predefined threshold. A label w_{ij} is associated with the link $l_{(S_i,S_j)}$, and is set to 0 if $\sum_{k=1}^{q} R_{jk} \leq (q - \delta_1)$; otherwise w_{ij} is 1. If the number of 0-labelled edges from S_i does not fall below a threshold δ_2 then S_i is detected to be fault-free, and the decision is broadcast. The remaining undetermined nodes can be detected later by their already detected fault-free neighbors after repetitively checking for $\sqrt{n}$ times, on average.

Peng Jiang [63] has advocated an improved fault detection scheme for WSNs.

Table 2.7: Neighbor Coordination Based Diagnosis Protocols - I

Protocol	Year	Key Points	Remarks
Krishnamachari and Iyengar [57]	2004	• Follows probabilistic estimation to detect faults. • Estimation is done taking number of matching neighbors.	• Concentrates on sensor measurement faults. • Results may be affected by unreliable sensor nodes.
Ding *et al.* [59]	2005	• Each sensor reading is compared with the median of neighbor's sensor readings. • If the deviation is extreme within a particular area, the node is considered faulty.	• If half of the neighbors are faulty, the algorithm cannot detect the faults..
Luo *et al.* [58]	2006	• Extension to the work in [57]. • Considers sensor faults along with measurement faults. • Minimizes communication overhead by selecting minimum neighbors, under a given detection error bound	• Generating global view at each not is not the objective. • Deals with event detection.
Chen *et al.* [60]	2006	• Threshold tests on measurement deviation among the neighboring sensors are performed at first. • Following majority voting on those test results, the actual fault status are deduced.	• Results for sparse networks may not be promising. • Cannot be used to detect temporary faults.
Hsin and Liu [61]	2006	• Each sensor actively monitors its neighbors. • Sensors also consult with the common neighbors before declaring a node as faulty.	• Simulations shown only for small networks (20 sensors). • Sensors are considered static during diagnosis.
Lee and Choi [62]	2008	• Copes with transient faults. • Measurement differences are tested for q rounds, followed by a majority voting protocol.	• Does not exploit the advantage of comparisons instead of reciprocal tests. • Not suitable for time-varying topology of the networks.
Peng Jiang [63]	2009	• Works similar to the protocol of Chen *et al.* [60]. • It follows a modified majority voting.	• If a node has no LG neighbors, pessimistically (may leads to false detection) it is considered to be fault-free (faulty) if the partial status is LG (LF). • Behavior with respect to large networks is not studied.
Choi *et al.* [64]	2009	• Extension to the work in [62]. • Adjusting the threshold by considering the effective node degree improves DA.	• Some nodes may remain undetermined that lowers the DA. • Time-varying topology in not considered.

Sensor S_i sets r_{ij} to 1, if v_{ij}^t exceeds the threshold δ_1. If it is not the case then $\Delta v_{ij}^{\Delta t}$ is calculated. If this difference exceeds the threshold δ_2 then r_{ij} is set to 1. In a similar way to the scheme of Chen *et al.* [60], this approach finds the partial status of the nodes, *i.e.*, LG or LF. For any sensor S_i, if the number of LG neighbors with test result 1 is not greater than half of the total number of LG neighbors, then S_i is detected fault-free; otherwise, S_i is considered faulty. If there is no LG neighbor of S_i then the status of S_i is changed to fault-free if it is LG before, or to faulty if it is LF before.

Choi *et al.* [64] have extended the work in [62] to adaptively adjust the parameters such as node degree and thresholds to improve the performance. After determining the faulty neighbors, a sensor node (say S_i) removes them from the neighbor table, reducing the effective node degree (d_i) that in turn reduces the value of the threshold (δ). For example, consider a node S_i with $d_i = 5$ and two of the neighbors are faulty. Here, $\delta = \lceil d_i/2 \rceil = 3$. If m_i is the number of matching neighbors of S_i then the condition, $m_i \geq \delta$ is satisfied and as a consequence S_i is detected fault-free. If one more neighbor becomes faulty then S_i cannot be detected as fault-free since the condition, $m_i \geq \delta$ is not satisfied with $m_i = 2$ and $\delta = 3$. However, dynamically adjusting d_i to 3, by removing the already detected faulty neighbors, updates the value of δ to 2. Subsequently, with two matching neighbors S_i can be detected as fault-free. However, in order to reduce the communication complexity, the authors have chosen to execute the last step maximum once to detect the fault status of the undetermined nodes. This may result in degraded accuracy. Consider the network as shown in Figure 2.2. The dark nodes are faulty. No other node except S_1 can pass the threshold test. So S_1 is detected as fault-free and the status of all other nodes are undetermined. If the last step of the algorithm is executed only once, then S_3 and S_7 are detected as faulty, and S_2, S_4, and S_8 are detected as fault-free, leaving the status of S_5, S_6, S_9, and S_{10} still undetermined.

An adaptive fault-tolerant event detection scheme for WSNs has been proposed by Yim *et al.* [17]. This approach can cope with transient faults. Fault and event detections are based on the confidence levels of a node on its neighbors. The confidence

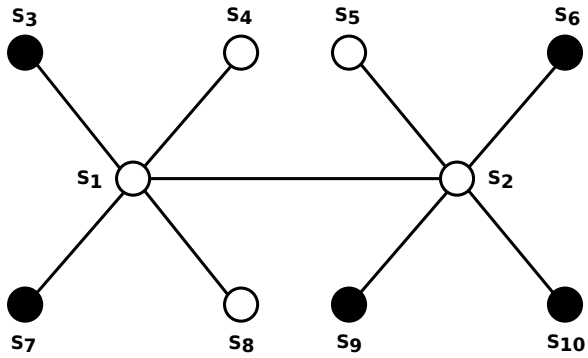


Figure 2.2: Illustration of diagnosis approach by Choi *et al.* [64]

levels are updated each time a fault or event is detected. The threshold for event detection is dynamically adjusted to improve the performance.

Miao *et al.* [65] have discussed a diagnosis mechanism called Agnostic Diagnosis (AD), exploiting the fact that the system metrics (such as radio-on time, number of packets transmitted, etc.) of sensors exhibit certain correlation patterns. Correlation graph that describes the latent status of the node is maintained and updated periodically. An abnormal node can be detected by examining the behavior of the correlation graph. Another variant of voting protocols called weighted-majority voting protocols can be used to detect faulty nodes. In these protocols, during coordination, the neighbors' decisions are considered with certain weights. The weights are defined with respect to different properties such as physical distance, confidence level, etc.

Sun *et al.* [66] have presented a confidence weighted voting (CWV) protocol to detect faulty sensor measurements. CWV gives more weights to the nodes that are more likely to be correct. The decisions of the nodes with more weights are more trustworthy, and results better detection accuracy.

Xiao *et al.* [67] have presented an in-network voting scheme to determine faulty sensor readings. In this approach, each node is assigned a *SensorRank* exploiting the correlations between readings of sensors. Considering these *SensorRanks* as weights, and following a trust voting protocol, it is decided whether to disregard a sensor reading or not.

Gao *et al.* [68] have proposed a distributed Weighted Median Fault Detection Scheme (WMFDS) for WSNs. Based on the confidence degree of the nodes, the

Table 2.8: Neighbor Coordination Based Diagnosis Protocols - II

Protocol	Year	Key Points	Remarks
Yim *et al.* [17]	2010	• Fault and event detections are based on the confidence levels on neighbors. • Follows adaptive threshold modification.	• Mainly focuses on event detection. • May result in degraded performance with mobile sensors during detection.
Miao *et al.* [65]	2011	• Exploits the fact that system metrics of sensors exhibit certain correlations. • Abnormal nodes can be detected by mining through these correlations.	• Deriving a consistent fault consensus at each node is deferred. • Noise related flaws are detected as faults.
Sun *et al.* [66]	2005	• Confidence levels are calculated based on similarity of measurements. • The decisions of nodes with more weight are considered more trustworthy.	• Neither can cope with dynamically moving hosts nor with temporary faults.
Xiao *et al.* [67]	2007	• Sensors are assigned ranks (weights) based on the correlation between the measurements. • Follows a trust voting protocol to determine faulty measurements.	• More iterations (to have a steady state of sensor ranks) increases the message complexity and diagnosis latency. • Order of execution of trust voting algorithm has an impact on final outcome.
Gao *et al.* [68]	2007	• Current sensor reading is compared with the weighted median of the neighbor sensor readings. • A node is declared faulty, if its confidence degree becomes 0.	• DA and FAR are affected by the initial confidence degree chosen.
Guo *et al.* [69]	2009	• Ranks are assigned considering the distances from the event, and the sensor measurements. • A node is considered faulty, if measurement rank and data rank difference is significant.	• Simulations are shown only for small networks (25 nodes). • For large and sparse networks the results may not be supportive.
Jhang *et al.* [70]	2009	• The similarity tests based on *extended Jaccard coefficient* helps in determining some fault-free nodes, initially. • The decisions of these fault-free nodes are used to diagnose the rest undetermined nodes.	• The approach does not outperforms other existing approaches that exploits spatio-temporal correlation of sensor readings.

weighted median of the neighboring sensor measurements is calculated and then, following a threshold based mechanism, the faulty nodes are detected.

Guo *et al.* [69] have presented FIND: a faulty node detection scheme for WSNs, considering data faults in sensors. It works on the basis that the sensor measurements change monotonically as the sensor nodes become farther from the event. The nodes are ranked on the basis of their sensor measurements as well as their physical distances from the event. A node is considered faulty, if there is a significant mismatch between the sensor data rank and the distance rank.

The neighbor coordination technique that has been suggested by Zhang *et al.* [70] uses extended Jaccard coefficient as a similarity function to compute the correlation degree between the neighboring nodes. The results of these similarity tests help in determining some fault-free nodes, in the first step. Based on the decisions of these fault-free nodes, the status of all other nodes are derived. The summary of neighbor coordination based diagnosis approaches is given in Table 2.7 and 2.8.

2.2.5 Hardware Based Approaches

In these approaches, the architecture of a node self-monitors to detect faults in it. Such approaches employ additional hardware at the node level.

The sensor nodes may become faulty due to battery exhaustion. If the hardware is capable of determining the current battery voltage [71, 72] then, by analyzing battery discharge curve and current discharge rate, such faults can be detected.

In order to detect physical malfunctions of sensor nodes caused due to impact or incorrect orientation, a flexible architecture has been proposed by Harte *et al.* [73]. A hardware circuit consisting of a number of accelerometers covers the sensor node. The accelerometers are useful to sense the physical condition of the node. Software modules developed on TinyOS operating system are used to analyze the data from the accelerometers to determine the orientation of the node and to detect impacts.

Sridharan *et al.* [74] have proposed an invariant based architecture to tolerate self faults in WSNs. The architecture is based on the *detectors* and *correctors* framework suggested by Arora and Theimer [75]. The specification and invariants of each component are set as a priori. Any violation of these invariants is considered fault. If the

violation count exceeds a given threshold, then the component, and hence the node, is considered faulty. Hardware based fault diagnosis protocols are briefed in Table 2.9.

Table 2.9: Hardware Based Diagnosis Protocols

Protocol	Year	Key Points	Remarks
Rakhmatov *et al.* [71]	2001	• Faults due to battery exhaustion can be handled by specialized hardware. • The hardware module can analyze the battery discharge curve and current discharge rate to detect such faults.	• Not applicable for temporary faults.
Harte *et al.* [73]	2005	• The accelerometers on the hardware circuit that covers the node helps in sensing the physical condition of the node. • The data from the accelerometers are analyzed by the software modules to determine orientation and detect impacts.	• The complexity of the node increases.
Sridharan *et al.* [74]	2008	• The violation of pre-specified invariants of any component is considered to be a fault. • The node is detected to be faulty, if the violation count exceeds cetain threshold.	• Energy consumption by the additional hardware is a key issue.

2.2.6 Neural Network Based Approaches

In these approaches, the nodes of a network are treated as the neurons of the corresponding neural network (NN). These neurons undergo training to learn the characteristics of the system, when the network is in a healthy state. A faulty node can be detected, if its characteristics deviate from the predicted characteristics by the NN.

Zhang *et al.* [76] have presented a fault diagnosis scheme for WSNs, based on a three layer radial basis function neural network (RBFNN). The top layer is the state recognition layer, in which the belief assignments for the sensors are obtained by using RBFNN. The measurements of two sensors, S_i and S_j, are the inputs to the RBFNN, that results in one output $m_{ij}(\{OK_i, OK_j\})$, or simply $m_{ij}(OK)$ which indicates that both S_i and S_j are fault-free. The outputs of the top layer are united into a common

frame, in the middle layer. The function of the bottom layer is evidence fusion and state decision.

Table 2.10: Neural Network Based Diagnosis Protocols

Protocol	Year	Key Points	Remarks
Zhang *et al.* [76]	2006	• Belief assignments for the sensors are obtained by using RBFNN with two inputs and one output. • Common frame is generated from the outputs of RBFNN, and then following evidence fusion the states of the sensors are decided.	• With increase in number of sensors, the combination method can be heavy to compute.
Jabbari *et al.* [77]	2007	• Residuals are generated by comparing the measured data with the network prediction. • These residuals are then analyzed by a probabilistic neural network to decide the status of the sensors.	• The complexity of training and analysis of large number of input combinations increases for large and dense networks.
Azzam and Rastko [78]	2008	• An *ad hoc* RNN is introduced considering the trust factors among the neighboring sensors. • The difference between the current sensor reading and the neural network output forms the basis of diagnosis.	• More sensors per node will increase the size of the *ad hoc* RNN and hence, the complexity.
Zhu *et al.* [79]	2010	• Standard prediction error (SPE) is calculated by comparing the actual values with the predicted values. • A faulty situation is detected if SPE increases suddenly.	• The number of faulty sensors is fixed. • For locating multiple sensor faults, it imposes a brute force method of considering all possible combinations of faulty sensors.

Jabbari *et al.* [77] have suggested a two-phase fault detection and recovery scheme. In the first phase (residual generation phase), a generalized regression neural network is used to generate the residuals by comparing the measured data with the network prediction. These residuals are analyzed in the second phase (residual verification phase), using a probabilistic neural network, to detect the faulty sensors.

Azzam and Rastko [78] have presented a modified recurrent neural network (RNN), called *ad hoc* RNN, to detect sensor faults. The *ad hoc* RNN considers the confidence

or trust factors ($0 < T_{ij} < 1$) between sensor nodes S_i and S_j. The modelling of the WSNs using *ad hoc* RNN can be divided into two phases: a learning phase, and a production phase. The weights corresponding to the fault-free and faulty nodes are adjusted by the neural network, in the learning phase. Following it, in the production phase, the current measurement of the sensor node is compared with the output of the neural network. If the difference is significant then the node is considered faulty.

Zhu *et al.* [79] have proposed a diagnosis model for WSNs based on principal component analysis (PCA) and neural network. PCA effectively reduces the size of the input data. Using previous knowledge, the predicted values of the sensors are computed. The square prediction error (SPE) for the sensor system is calculated by comparing the actual measurements with the predicted values. If SPE suddenly increases, then a faulty situation is detected. Considering different combinations of sensors, the SPE is recalculated after replacing the actual measured values with the predicted values. The new SPE value is used to locate the faulty sensors. Neural network based approaches for fault diagnosis in wireless networks are summarized in Table 2.10.

2.3 Summary

The inherent challenges in wireless networks and their application in many safety critical scenarios mandates it to devise effective fault diagnosis algorithms with high accuracy, low false alarm rate, and less communication overhead. The observations from the *as-is* analysis of the existing diagnosis protocols can be put into the following bullet points.

- Few work has been done on fault diagnosis in MANETs.
- The majority of existing protocols are developed on the basis of exchange of diagnostic messages. High communication overhead affects the performance of the wireless networks, especially when they operate with limited energy.
- The more convoluted intermittent and transient faults have not been investigated much.

- Most of the protocols constrain node's mobility so that the network topology remains static during diagnosis.
- Since transient faults are caused due to external errors and are often considered fault-free, it is highly important to discriminate transient from intermittent faults.

Though quite a good number of fault diagnosis algorithms are in place, these observations provide enough scope to improve the diagnosis performance. These problems are studied more vigilantly in the subsequent chapters.

Chapter 3

Fault Diagnosis in Static Topology MANETs

Fault diagnosis in MANETs considering static topology of the network has been studied in [8, 10, 32]. Due to the adoption of spanning tree based dissemination strategy, instead of a flooding based one, the communication complexities of dynamic and adaptive DSDPs [8, 32] are observed to be less as compared to static DSDP [10]. However, in these protocols, a node is required to respond to $(\sigma + 1)$ received test requests, where σ is the connectivity of the network. This leads to higher computation cost, and communication complexity. This aggravates further for denser networks. The case is even worse for static DSDP, where each node responds to every incoming test request.

This chapter presents a DSDP to handle permanent faults in static topology MANETs, where each node is required to respond to only one test request from its neighbors. The proposed DSDP exploits the fact that responding to one test request and following the shared nature of communication of ad hoc networks, it is possible to detect the actual status of a node by at least one fault-free neighbor. This not only reduces the communication complexity, but also the computation cost. Another advantage of having less message exchanges is the high fault detection accuracy, due to less interference and message loss.

The rest of this chapter is organized as follows. Section 3.1 explains the proposed fault diagnosis protocol. Section 3.2 demonstrates the analytical study to prove the correctness and completeness of the proposed protocol. The experimental results

are shown in Section 3.3 that vows the effectiveness of the protocol. The work is summarized in Section 3.4.

3.1 Proposed DSDP

The proposed protocol is developed on the top of comparison-based model. It executes in two phases: *Testing* phase, and *Dissemination* phase. During the *Testing* phase, a unit can test its neighbors by sending them test tasks, along with it can reply to the received test message. This phase starts periodically by the initiator, and other nodes are subsequently triggered to initiate their diagnosis. Once the test request has been sent, each mobile gathers the responses from its neighbors based on which they can be diagnosed. This phase terminates as a predefined time-out period expires. At the end of this phase, every fault-free mobile has the status of a subset of hosts in the network, called a *local diagnosis view*. During the *Dissemination* phase, the local diagnosis views are exchanged between the hosts to generate the global view of the network at each fault-free node.

Flooding based approach is a trivial information exchange approach in WSNs. However, the biggest deficiency of this approach that makes it unsuitable for WSNs is *message implosion* problem [80, 81]. To overcome this, several efficient flooding mechanisms and improvements have been proposed [82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95]. Threshold based flooding improvements (counter-based, distance-based, and location-based) are discussed in [82, 83], where nodes decide whether to rebroadcast the flooding packet or not, depending on certain thresholds. In self-pruning, dominant pruning [84] and neighbor-coverage scheme [83], a node relay received packet only if it has neighbors not covered by previous forwarding nodes. Multipoint Relay (MPR) scheme [85, 86, 87, 88, 89] improves flooding efficiency by choosing a minimal set of neighbors, that covers all 2-hop neighbors of a node, for flooding. In [90], authors have discussed a Topology Broadcast based on Reverse-Path Forwarding (TBRPF) scheme where each node generates spanning tree (also called source tree) providing paths to all reachable nodes. Only the non-leaf nodes in the source tree, where the root is the source node, rebroadcasts the flood packet. However,

maintenance of multiple source trees is complex. In cluster based flooding schemes [82], only the cluster heads and gateway nodes act as forwarding nodes. Ogier *et al.* [91] have discussed a Connected Dominating Set (CDS) based flooding in which MANET Designated Routers (MDRs) are responsible for flooding link state advertisements (LSAs). Adjacencies are formed only between MDRs or Backup MDRs and a subset of their neighbors to reduce flooding overhead. Some MDRs serve as Backup MDRs which provide flooding when neighboring MDRs fail. The set of MDRs forms a connected dominating set , and the set of MDRs and Backup MDRs forms a bi-connected dominating set. Baccelli *et al.* [92] have discussed an overlay subgraph based on a Relative Neighbor Graph (RNG) scheme called Synchronized Link Overlay - Triangular (SLOT), with the objective to have low overlay link density, and low overlay link change rates. Authors have claimed SLOT to be efficient with respect to control traffic; thus useful in dense networks. Cordero *et al.* [93] have discussed the impact of jitter on flooding that reduces the number of packet collisions and the number of transmissions, but with increased delay. Although these approaches are improvements to blind flooding technique, but would incur more communication cost to achieve the objective of the proposed algorithm, *i.e.*, to disseminate the local diagnostic views for generating global view at each fault-free node in the network. Another efficient dissemination scheme called RPL (Routing Protocol for Low-Power and Lossy Networks) is the key discussion in [94]. RPL organizes the topology as a set of one or more Destination Oriented Directed Acyclic Graphs (DODAGs) that are used for message dissemination.

The generation, and dissemination of global diagnosis view among fault-free nodes in the network requires a connected topology among them. This is possible with minimum $(n-1)$ number of edges, where n is the number of fault-free nodes. Therefore, we adapt a spanning tree (ST) based dissemination mechanism [8, 11], similar to DODAGs in RPL, to have less communication complexity for dissemination. A distributed ST, rooted at a robust node (*initiator*), that overlays all fault-free nodes in the network is constructed during *building phase*. Here, the term *distributed* means no sensor has the complete ST information, rather each node in the tree keeps the

information about its parent and the list of children. The self diagnosis session terminates once the global diagnostic is propagated, through the constructed ST, to all fault-free mobiles in the network.

These phases are described more clearly in the following sub sections, in sequel.

3.1.1 Testing Phase

In this phase, messages of type TestResponse are communicated among the nodes to perform tests, and gather responses. The diagnosis session is started by the initiator by sending a test task $T_{initiator}$ to its neighbors.

A mobile node S_i deals with the following types of events in this phase.

- *Test Generation* − A mobile S_i may wish to test its neighbors because it has received a TestResponse message for the first time that intimates it about the initiation of the diagnosis session. If S_i is the initiator then it starts the diagnosis session periodically. Node S_i generates a test task, say T_i, that is required to be executed by all its neighbors.

- *Response Generation* − Upon receiving the first TestResponse message, say from neighbor S_j, mobile S_i generates a reply with the outcome of the test task T_j. This output is piggybacked with the test generated by node S_i and the whole message is broadcast to all its neighbors. So node S_i broadcasts $m_{S_i} = (\text{TestResponse}, T_i, S_j, T_j, O_j^{S_i})$. Here, T_i is the test generated by the mobile S_i, and $\{S_j, T_j, O_j^{S_i}\}$ constitutes the response by S_i to the test task T_j sent by neighbor S_j. Moreover, the TestResponse message is sent only if it has not already responded to any other test. At this point, a timer is activated and set to T_{out}, which is chosen such that the slowest neighbor that is not hard faulty is able to execute the task, and respond within that time bound. Thus, the time-out mechanism is helpful to detect the hard faulty neighbors. A response set $Validated_{S_i}$, is maintained to keep track of all correct test responses, either generated by the node itself or deduced during the diagnosis. Mobile S_i maintains another set $Pending_{S_i}$, to store the responses for which it is unable to classify as correct or wrong.

- *Reception of task result* – Assuming that mobile S_i has received a TESTRESPONSE message that includes the response set $\{S_k, T_k, O_k^{S_j}\}$ from $S_j \in N_{S_i}$, three different scenarios may arise.

  Scenario 1: $\exists < T_k, R > in\ Validated_{S_i}$ – In this case, mobile S_i has already validated the correct result R of T_k, received from some other neighbor. So S_i can diagnose the status of S_j by simply comparing the result $O_k^{S_j}$ with R. If the comparison passes, S_j is added to the fault-free list of S_i, *i.e.*, $FF_{S_i} = FF_{S_i} \cup \{S_j\}$; otherwise, S_j is detected faulty and is added to the faulty list of S_i, *i.e.*, $F_{S_i} = F_{S_i} \cup \{S_j\}$.

  Scenario 2: $\exists < T_k, O_k^{S_z} > in\ Pendings_{S_i}$ – In this scenario, S_i has already received the result of the same task T_k from $S_z \in N_{S_i}$. So S_i can now compare both the outputs. If $O_k^{S_j}$ and $O_k^{S_z}$ are the same, then both nodes S_z and S_j are diagnosed to be fault-free. Consequently, the fault-free set of S_i is updated, *i.e.*, $FF_{S_i} = FF_{S_i} \cup \{S_z, S_j\}$, and $< T_k, O_k^{S_z} >$ is removed from the pending list, *i.e.*, $Pending_{S_i} = Pending_{S_i} - \{< T_k, O_k^{S_z} >\}$. At this point S_i validates the correct result of T_k, and hence puts $< T_k, O_k^{S_j} >$ in its validated list, *i.e.*, $Validated_{S_i} = Validated_{S_i} \cup \{< T_k, O_k^{S_j} >\}$. However, if $O_k^{S_j}$ and $O_k^{S_z}$ are different then node S_j cannot be diagnosed at this point and, hence, is added to the pending list, *i.e.*, $Pending_{S_i} = Pending_{S_i} \cup \{< T_k, O_k^{S_j} >\}$.

  Scenario 3: $\nexists < T_k, R > in\ Pending_{S_i}$ – In this case, mobile S_i has received the first response corresponding to either its test request or one of its neighbor's test requests. Mobile S_i can execute the task that is received along with the output, in order to be able to diagnose the status of the mobile from which it has received the response. But, if S_i has more than one outputs received for the same task from different neighbors, then a simple asymmetric comparison model may be followed. Therefore this step is performed only after T_{out} expires, and no more response is expected. The first test response received for any task T_k is stored in the pending list, *i.e.*, $Pending_{S_i} = Pending_{S_i} \cup \{< T_k, O_k^{S_j} >\}$.

Algorithm 3.1: Testing Phase

Data: $C(S, L)$: Communication graph of MANET.
Result: Local diagnosis view at each node in the MANET.
//Responded−Boolean variable initially FALSE; set to TRUE once a node sends a response.

1 msg = Message received from mobile $S_j \in N_{S_i}$;
2 **switch** $msg.Type()$ **do**
3 **case** TestResponse*: // i.e.,* $msg = (\text{TestResponse}, T_j, S_k, T_k, O_k^{S_j})$
4 **if** *!Responded* **then**
5 Compute $O_j^{S_i}$ = Output of test task T_j;
6 **if** $S_i == initiator$ **then**
7 $\_\text{rb}(m_{S_i} = (\text{TestResponse}, \phi, S_j, T_j, O_j^{S_i}))$;
8 **else**
9 generate test task T_i;
10 $\_\text{rb}(m_{S_i} = (\text{TestResponse}, T_i, S_j, T_j, O_j^{S_i}))$;
11 Set timer to T_{out};
12 **end**
13 Responded=True;
14 **end**
15 **if** $\exists < T_k, R > in\ Validated_{S_i}$ **then**
16 **if** $O_k^{S_j} == R$ **then**
17 $FF_{S_i} = FF_{S_i} \cup \{S_j\}$
18 **else**
19 $F_{S_i} = F_{S_i} \cup \{S_j\}$
20 **end**
21 **else if** $\exists < T_k, O_k^{S_z} > in\ Pendings_{S_i}$ ***and*** $O_k^{S_j} == O_k^{S_z}$ **then**
22 $FF_{S_i} = FF_{S_i} \cup \{S_j, S_z\}$;
23 $Pendings_{S_i} = Pendings_{S_i} - \{< T_k, O_k^{S_z} >\}$;
24 $Validated_{S_i} = Validated_{S_i} \cup \{< T_k, O_k^{S_j} >\}$;
25 **else**
26 $Pendings_{S_i} = Pendings_{S_i} \cup \{< T_k, O_k^{S_j} >\}$;
27 **end**
28 **end**
29 **case** TimeOut*: // i.e., the delay* T_{out} *has expired*
30 **for** ***each*** $< T_k, O_k^{S_z} > \in Pendings_{S_i}$ **do**
31 **if** $\exists < T_k, R > in\ Validated_{S_i}$ **then**
32 **if** $O_k^{S_z} == R$ **then**
33 $FF_{S_i} = FF_{S_i} \cup \{S_z\}$;
34 **else**
35 $F_{S_i} = F_{S_i} \cup \{S_z\}$;
36 **end**
37 **else**
38 Compute $O_k^{S_i}$ = Output of task T_k;
39 **if** $O_k^{S_i} == O_k^{S_z}$ **then**
40 $FF_{S_i} = FF_{S_i} \cup \{S_z\}$;
41 **else**
42 $F_{S_i} = F_{S_i} \cup \{S_z\}$;
43 **end**
44 $Validated_{S_i} = Validated_{S_i} \cup \{< T_k, O_k^{S_i} >\}$;
45 **end**
46 **end**
47 $F_{S_i} = F_{S_i} \cup \{N_{S_i} - (FF_{S_i} \cup F_{S_i})\}$;
48 **end**
49 **end**

- TimeOut − As the timer expires, indicated by the reception of a TimeOut message, mobile S_i processes its pending list of responses. For a pending response $< T_k, O_k^{S_z} >$, if there exists a matching entry, say $< T_k, R >$, then the status of S_z can be detected by comparing $O_k^{S_z}$ with R. If no match is found in $Validated_{S_i}$ then S_i explicitly executes the task T_k and the obtained result is compared with $O_k^{S_z}$ to determine the fault status of S_z. The set of neighbors that did not reply within the time bound T_{out} are diagnosed faulty, and are included in the set of faulty nodes maintained by S_i, *i.e.*, $F_{S_i} = F_{S_i} \cup \{N_{S_i} - (FF_{S_i} \cup F_{S_i})\}$. At this point mobile S_i has the status of all its neighbors constituting the local diagnosis view.

Steps followed in this phase can be precisely put into Algorithm 3.1.

3.1.2 Dissemination phase

In this phase, three different types of messages are exchanged among the fault-free mobiles, *viz.*, SpanTree, LocalDiagnostic, and GlobalDiagnostic messages; and the following events are handled.

Algorithm 3.2: Build Spanning Tree

Data: $C(S, L)$: Communication graph of MANET.
Result: Spanning Tree covering all fault-free nodes.

```
1 msg = SpanTree message from S_j ∈ N_{S_i} // i.e., m_{S_j} = (SpanTree, Parent_{S_j})
2 if S_j is found fault-free in VIEW_{S_i} then
3 |   if Parent_{S_i} == φ and S_i ≠ initiator then
4 |   |   Parent_{S_i} = S_j;
5 |   |   _rb(SpanTree, Parent_{S_i})
6 |   else if S_i == Parent_{S_j} then
7 |   |   Children_{S_i} = Children_{S_i} ∪ {S_j};
8 |   end
9 end
```

- *Build Spanning Tree* − Messages of type SpanTree are exchanged to construct a ST covering all fault-free nodes. This is initiated by a robust node called *initiator*. In general, a node S_j transmits $m_{S_j} = (\text{SpanTree}, Parent_{S_j})$. If S_j is the initiator then $Parent_{S_j} = \phi$. Node S_i, upon receiving the first SpanTree message from $S_j \in N_{S_i}^t$, verifies from its local view if S_j is fault-free. After

confirmation, S_i adds S_j in its children set ($Children_{S_i}$) if S_j has already chosen S_i as its parent, *i.e.*, $S_i = Parent_{S_j}$; otherwise, S_i chooses fault-free S_j as its parent in the ST, and intimates the same to its neighbors by sending its own SpanTree message. These steps are more precisely given in Algorithm 3.2. The current diagnostic view of S_i, denoted as $VIEW_{S_i}$, is composed of FF_{S_i} and F_{S_i}.

- *Dissemination* – Two types of messages, LocalDiagnostic and GlobalDiagnostic, are exchanged to generate global view at each fault-free node. Once the ST is constructed, all leaf nodes start disseminating their local diagnostics to their parents.

Algorithm 3.3: Dissemination

Data: $C(S, L)$: Communication graph of MANET.
ST: Spanning Tree.
Result: Each fault-free node S_i has the global diagnostic view.

```
1  if Children_{S_i} == φ then
2  |  _rb(m_{S_i} = (LocalDiagnostic, VIEW_{S_i}));
3  end
4  msg = Message received from mobile S_j ∈ N_{S_i};
5  switch msg.Type() do
6  |  case LocalDiagnostic: // i.e., msg = (LocalDiagnostic, VIEW_{S_j})
7  |  |  if S_j ∈ Children_{S_i} then
8  |  |  |  Update VIEW_{S_i};
9  |  |  |  ChildrenSentLD_{S_i} = ChildrenSentLD_{S_i} ∪ {S_j};
10 |  |  end
11 |  |  if ChildrenSentLD_{S_i} == Children_{S_i} then
12 |  |  |  if S_i ≠ initiator then
13 |  |  |  |  _rb(m_{S_i} = (LocalDiagnostic, VIEW_{S_i}));
14 |  |  |  else
15 |  |  |  |  _rb(m_{S_i} = (GlobalDiagnostic, VIEW_{S_i}));
16 |  |  |  end
17 |  |  end
18 |  end
19 |  case GlobalDiagnostic:
20 |  |  if S_j == Parent_{S_i} then
21 |  |  |  Update VIEW_{S_i};
22 |  |  |  _rb(m_{S_i} = (GlobalDiagnostic, VIEW_{S_i}));
23 |  |  end
24 |  end
25 end
```

A mobile node S_i handles the diagnostic messages as described below. The steps are shown clearly in Algorithm 3.3.

- LocalDiagnostic – After receiving such message from its child S_j, *i.e.*, $msg = (\text{LocalDiagnostic}, VIEW_{S_j})$, node S_i updates its view to include the local view of S_j. Each mobile S_i maintains a set $ChildrenSentLD_{S_i}$, to keep track the list of children those have already sent their local diagnostics. S_i disseminates its local diagnostic only when all its children have sent their local diagnostics to it. However, if S_i is the initiator then it has received the local views of all fault-free nodes, and deduced the global view. The initiator then starts disseminating the global view down the ST.
- GlobalDiagnostic – If S_i receives a GlobalDiagnostic message from its parent S_j then it updates its view to incorporate the global view, and subsequently broadcasts it to its children if it is a non-leaf node.

3.2 Analytical Study of Proposed DSDP

The diagnosis protocol is viable iff it is *correct* and *complete*. These two feasibility properties are defined in Section 1.5. In this section, the proofs of correctness and completeness of the proposed DSDP are given through analytical derivations.

3.2.1 Correctness of the proposed DSDP

The correctness of the protocol can be defined with respect to *correct partial local diagnosis* and *correct dissemination* of local and global diagnosis information. If a fault-free mobile S_i deduces the state of all its neighbors correctly at the end of the *Testing* phase, then S_i is said to have correct partial local diagnosis view. During the *Dissemination* phase, if local views of every fault-free unit reaches the root and subsequently the global view at the root is propagated to every fault-free node in the ST then correct dissemination is achieved.

We define the following lemmas to support the correctness and completeness of the proposed DSDP.

Lemma 3.1. *Let S, the set of nodes; and L, the set of logical links; collectively define a connected graph $C(S, L)$ that represents the communication graph of a MANET, and F be the set of faulty units at the time of diagnosis. Every node $S_i \in S$ has at least one fault-free neighbor, i.e., for every node $S_i \in S$, $\exists\ S_j$ in N_{S_i} such that $S_j \notin F$.*

Proof. The maximum number of faults that a σ-diagnosable system can tolerate is given by $(\xi - 1)$, *i.e.*, $\sigma < \xi$ [8], where ξ is the node-connectivity of the graph C. This condition directly stems from the fact that if $\sigma \geq \xi$ then removal of ξ faulty nodes from C may result in a disconnected or trivial graph. Furthermore, the minimum number of neighbors of a node S_i is ξ, *i.e.*, $|N_{S_i}| \geq \xi$, due to the same reason as above. It implies that $|N_{S_i}| > \sigma$. Hence, there is at least one unit $S_j \in N_{S_i}$ which is fault-free. □

Lemma 3.2. *Let the connected graph $C(S, L)$ represents a fixed topology MANET, where S and L respectively represents the set of mobiles and the set of logical links between them. At the end of Testing phase, the status of any node $S_j \in S$ is correctly diagnosed by all units $S_i \in N_{S_j}$ that are fault-free.*

Proof. Based on the proposed protocol, any node $S_j \in S$ responds to only one test task. However, it transmits the result along with the task for which the result is generated to all $S_i \in N_{S_j}$, *i.e.*, its TestResponse message includes $\{T_k, O_k^{S_j}\}$, where T_k is the first test task received and $O_k^{S_j}$ is the output of T_k generated by node S_j. If S_i is fault-free then it can follow an asymmetric comparison model to compare the result $O_k^{S_j}$ with similar results, say $O_k^{S_z}$ (if any), for the same task received from some other unit $S_z \in N_{S_i}$. However, if no matching output is found, then S_i itself executes the test task T_k and compares the produced result $O_k^{S_i}$ with $O_k^{S_j}$ to diagnose the state of S_j correctly, after the timer T_{out} expires. Moreover, if S_j is hard faulty then S_i detects this as faulty after the predefined timer expires. So in either case, S_i is able to deduce the state of mobile S_j properly. This is true $\forall S_i \in N_{S_j}$ that are fault-free. □

Lemma 3.3. (*Correct local diagnosis view*) *Let a fixed topology MANET is represented by the graph $C(S, L)$, where S is the set of mobiles and L is the set of logical links between them. At the end of the Testing phase, if $S_i \in S$ is fault-free then ($FF_{S_i} \cup$*

$F_{S_i}) = N_{S_i}$, *where* FF_{S_i} *and* F_{S_i} *denote respectively, the set of fault-free units and set of faulty units diagnosed by* S_i. *Also, for each mobile* $S_j \in N_{S_i}$, *if* S_j *is fault-free then* $S_j \in FF_{S_i}$, *and if* S_j *is faulty then* $S_j \in F_{S_i}$.

Proof. Any fault-free node $S_i \in S$ receives response from all units $S_j \in N_{S_i}$ which are not hard faulty. Lemma 3.2 states that S_i correctly diagnoses the state of all fault-free neighbors. Furthermore, if S_j is soft faulty then it will not pass any comparison followed by S_i during the *Comparison* phase, and hence it is never misdiagnosed as fault-free. The neighbors, those don't respond at all are declared faulty after T_{out} expires. So any unit $S_j \in S$ is either diagnosed as hard faulty, soft faulty, or fault-free by unit S_i correctly. Therefore, if unit S_j is fault-free then $S_j \in FF_{S_i}$, otherwise $S_j \in F_{S_i}$. Hence, $(FF_{S_i} \cup F_{S_i}) = N_{S_i}$. □

Lemma 3.4. *Let* S *be the set of mobiles,* L *be the set of logical links between them, and* $C(S, L)$ *be the communication graph of a* σ*-diagnosable MANET. The ST constructed during Dissemination phase overlays all and only fault-free nodes in the MANET.*

Proof. Lemma 3.1 asserts that each node has at least one fault-free neighbor. This implies that the graph C', generated from C by considering $(S - F)$ node set is also connected. Here, F is the set of faulty nodes in the network, and $|F| \leq \sigma$. According to Lemma 3.3, local diagnosis view of each fault-free node S_i, *i.e.*, $VIEW_{S_i}$ is correct. If a fault-free node S_i receives SpanTree message from a fault-free neighbor S_j, and S_i is not connected to the ST then it chooses S_j as its parent. Node S_j then conveys this event by sending its own SpanTree message. Upon receiving this message, S_i includes S_j in its children list, after confirming from $VIEW_{S_i}$ that S_j is fault-free. Thus a fault-free node is always connected to a fault-free parent, and it has only fault-free children in the ST. Hence, the ST overlays all and only nodes in $(S - F)$. □

Lemma 3.5. (*Correct dissemination*) *Let* $C(S, L)$ *represents the connected communication graph for a fixed topology MANET, and* F *be the set of faulty units such that* $|F| \leq \sigma$. *During Dissemination phase, the local diagnosis view of every fault-free node* $S_i \in S$ *correctly propagates to the root of the ST. Subsequently, the global view at the root node is correctly disseminated to all fault-free nodes in the network.*

Proof. During the *Dissemination* phase, leaf nodes in the ST start disseminating their local views to their parents. The parent nodes collect the local views from their children. Then they send the aggregated view up in the tree. Hence, the local diagnosis views of all the nodes in the ST reaches the root node. Moreover, the ST constructed during *Dissemination* phase covers all fault-free hosts in the MANET (Lemma 3.4). Hence, the root receives the local views of each fault-free mobile $S_i \in S$. The global view generated at the root node is then disseminated to its children, which is then relayed to the nodes in the subsequent levels in the tree. In this way, correct dissemination of the global view is achieved. □

Theorem 3.1. (*Proposed DSDP Correctness*) *For a given connected graph* $C(S, L)$ *that represents a fixed topology MANET, let* F, F_{S_i}*, and* FF_{S_i} *respectively denotes the actual set of faulty units in the system at the beginning of the diagnosis session, the set of faulty units, and the set of fault-free ones deduced by mobile* S_i *at the end of diagnosis session. The proposed DSDP produces correct set of faulty and fault-free units, i.e., at the end of diagnosis session, no fault-free unit* $S_i \in S$*, misdiagnoses any node* $S_j \in F$ *as fault-free, or any node* $S_k \in (S - F)$ *as faulty.*

Proof. The proof of this theorem directly follows from Lemma 3.3 and 3.5. Lemma 3.5 states that the local diagnosis view of every fault-free mobile $S_i \in S$ is transmitted up in the ST to reach the root, correctly. The root node, then can deduce the global diagnosis view, which is correct. The correctness of the global diagnosis view follows from the correctness of local views (Lemma 3.3). So in the global diagnosis view, generated at the root node, neither any node $S_j \in F$ is detected as fault-free, nor any node $S_k \in (S - F)$ is determined faulty. The correct global diagnosis view is then disseminated down the tree to every node to realize the correct global view at each fault-free node in the network. □

3.2.2 Completeness of the proposed DSDP

The proof of completeness of the proposed DSDP can be derived from Lemma 3.3 and 3.5. This can be comprehended into Theorem 3.2.

Theorem 3.2. (*Proof of Completeness*) *In an undirected graph $C(S, L)$, that represents the communication graph of MANET, where S and L subsequently represents the set of hosts and the set of logical links among them, if F_{S_i}, and FF_{S_i} are the set of faulty hosts, and set of fault-free ones deduced by $S_i \in (S - F)$ at the end of diagnosis session, then $(F_{S_i} \cup FF_{S_i}) = S$. Here F is the set of actual faulty units in the network.*

Proof. For any fault-free node S_i, before dissemination starts, $(F_{S_i} \cup FF_{S_i}) = N_{S_i}$. This can easily be verified from Lemma 3.3. Since correct dissemination is achieved, as proved in Lemma 3.5, the status of each node is propagated to each and every fault-free node in the network. Hence, at the end of the diagnosis session, each fault-free node S_i confirms a complete global view of the network, *i.e.*, $(F_{S_i} \cup FF_{S_i}) = S$. □

Theorem 3.3. (*Communication Complexity*) *The communication complexity of the proposed protocol is $O(n)$, where n is the number of nodes in the network.*

Proof. The proposed protocol involves two phases: a testing phase followed by a dissemination phase. The communication overhead in each of these phases can be found as follows.

(i) Testing phase − Each mobile, except the initiator, generates exactly one TestResponse message, whereas the initiator generates this message twice. Hence total number of TestResponse message exchanges is $(n + 1)$.

(ii) Dissemination phase − In this phase, following three types of messages are exchanged.

- SpanTree − Each mobile generates a SpanTree message in the worst case, i.e., when all the mobiles are fault-free. So it requires n such message exchanges to completely build the spanning tree.
- LocalDiagnostic − One LocalDiagnostic message is transmitted by every fault-free mobile except the root. So in the worst case, $(n-1)$ number of such messages are to be exchanged.
- GlobalDiagnostic − The protocol requires $(n-1)$ number of GlobalDiagnostic message exchanges. This happens in the worst case scenario

when all the nodes are fault-free and each level of the spanning tree contains exactly one node, i.e., the depth of the spanning tree is $(n-1)$.

Thus the total number of message exchanges required in the proposed diagnosis protocol is $(4n-1)$, and the communication complexity is $O(n)$. □

3.3 Simulation and Results

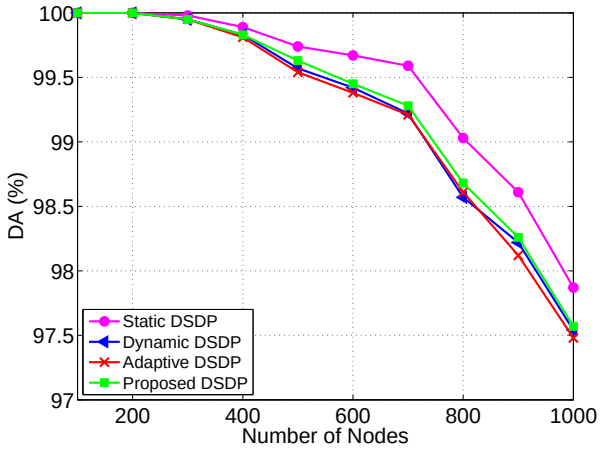
In this section, the performance of the proposed DSDP is evaluated . In order to study the behavior of the proposed protocol, a set of simulations is conducted using the INET framework on OMNeT++ simulator. The results obtained are compared with that of static DSDP [10], adaptive DSDP [32], and dynamic DSDP [8]. Three important performance measure criteria: *DA*, *FAR*, and *message complexity* are considered for discussion. The results obtained for different simulation scenarios are presented below.

3.3.1 Simulation scenario 1

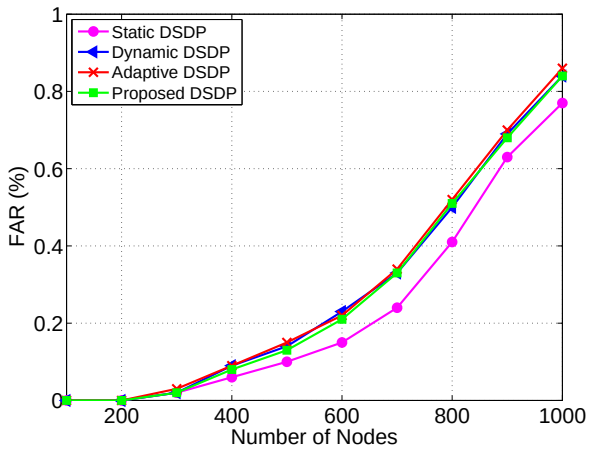
The first simulation scenario is set up considering 10% faults in the network. The network size is varied from 100 nodes to 1000 nodes, at an increment of 100. The obtained results for DA, FAR, and number of messages exchanged during the diagnosis period are depicted graphically in Figure 3.1.

3.3.2 Simulation scenario 2

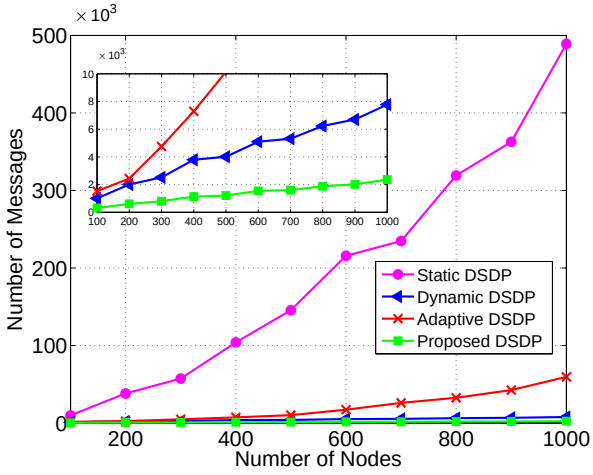
In this simulation scenario, the fault probability of a node is considered 0.2, *i.e.*, 20% of the nodes in the network are considered faulty. Accordingly, the values for DA, FAR, and communication complexity are recorded for varying sized networks; from smaller networks with 100 nodes to larger networks with 1000 nodes. Figure 3.2 shows the recorded results.



(a)

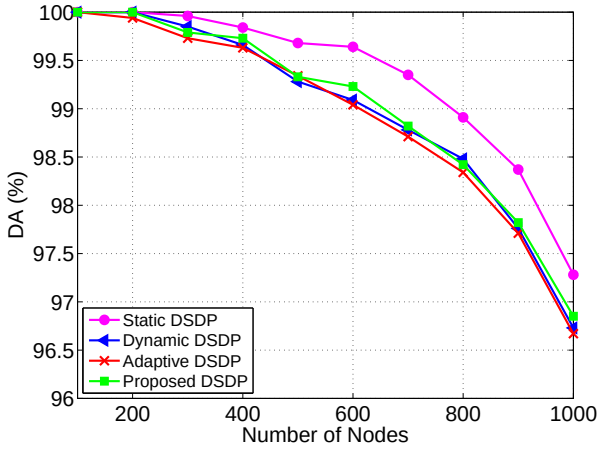


(b)

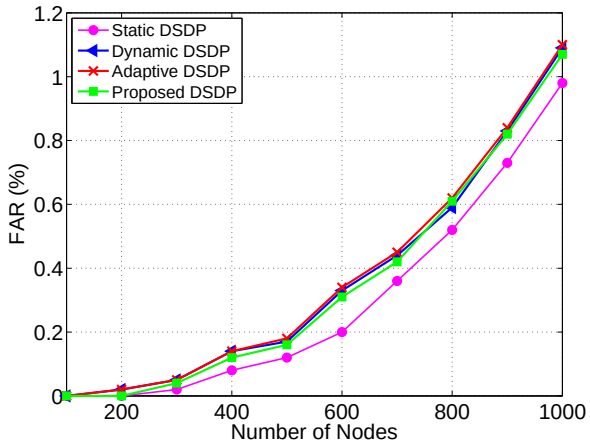


(c)

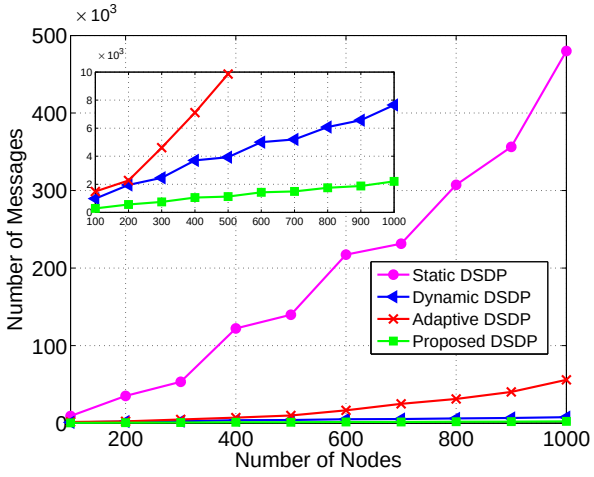
Figure 3.1: Behavior of (a) DA (b) FAR, and (c) Message complexity, considering 10% faults and varying network size.



(a)



(b)



(c)

Figure 3.2: Behavior of (a) DA (b) FAR, and (c) Message complexity, considering 20% faults and varying network size.

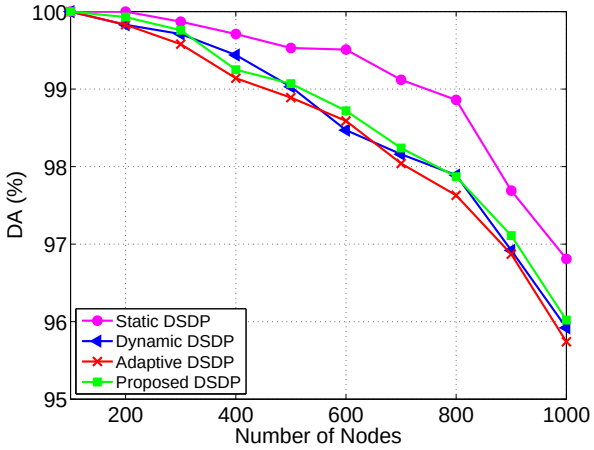
3.3.3 Simulation scenario 3

Keeping the percentage of the faulty nodes in the network fixed at 30%, this simulation scenario studies the behavior of DA, FAR, and message complexity with respect to increase in network size. The minimum and maximum number of nodes in the network are considered 100 and 1000 respectively. The behavior of the evaluation parameters in this simulation scenario are put in Figure 3.3.

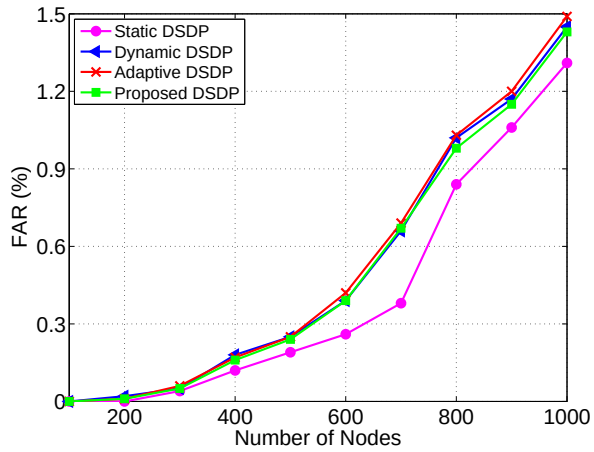
3.3.4 Discussion

In case of static DSDP, a node is required to respond to each and every incoming test request. In addition, due to the flooding based dissemination process adapted in this protocol the communication complexity is observed to be very high, which is not suitable for certain MANET applications. Dynamic, adaptive, as well as the proposed diagnosis protocol use a ST based dissemination process. Moreover, in case of dynamic and adaptive protocols, a node is required to reply $(\sigma + 1)$ test requests, where σ is the diagnosability of the network. This reduces the message traffic compared to static DSDP. Each node responds to only one test request in the proposed DSDP, which further reduces the message communication. Moreover, the proposed DSDP makes this independent of σ. It can also be noted that the communication complexity of adaptive DSDP is more as compared to dynamic and proposed protocol, even though all of them use similar dissemination scheme. This is due to the fact that, it requires less number of message communications to build the ST than to maintain it in case of adaptive DSDP. For more details SelfMaintainingPhase procedure in [32] may be referred. The vertical axis is scaled down in the sub graphs of Figure 3.1 (c), 3.2 (c), and 3.3 (c) to show the comparison of the number of messages exchanged during diagnosis in case of adaptive, dynamic, and proposed protocols more clearly.

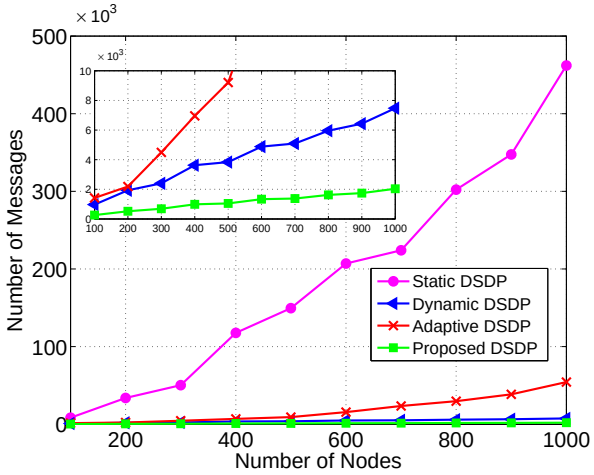
The detection accuracy of any fault diagnosis protocol depends on how much diagnostic information reaches to a particular node. More the diagnostic information received by a node, more is the detection accuracy. However, during the dissemination of local diagnostics there is a probability of message/information loss due to interference, and this aggravates as the network size increases. The DA and FAR in



(a)



(b)



(c)

Figure 3.3: Behavior of (a) DA (b) FAR, and (c) Message complexity, considering 30% faults and varying network size.

dynamic, adaptive, and proposed DSDP are comparable since they follow the same mechanism to disseminate local and global diagnostics. It is clearly seen from Figure 3.1, 3.2, and 3.3 that static DSDP observes more DA and less FAR compared to other diagnosis protocols under consideration. The reason being the availability of the number of connected paths between two fault-free mobiles. The diagnostic information of a node can reach another node following any of these connected paths. However, in case of ST based approaches, there exists only one path from any node to the root. So the message loss, in these protocols, costs more with respect to loss of information as compared to flooding based static DSDP.

As expected, increase in fault percentage results in decreased DA and increased FAR. The communication traffic decreases with number of faulty nodes in the network because of the involvement of lesser number of fault-free hosts during dissemination.

3.4 Summary

This chapter presented a distributed self diagnosis protocol that enables each fault-free mobile to identify the correct fault status of all units in the MANET. It considers both hard and soft faulty mobiles. In the proposed protocol, a node is required to respond to only one incoming test request. In addition to this, it uses a ST based approach to disseminate the local and global diagnosis views that results in reduced communication complexity. The static DSDP results in better detection accuracy with low FAR as compared to other protocols because of the adapted flooding based dissemination approach. The DA and FAR of adaptive, dynamic, and proposed diagnosis protocol are comparable. The proposed DSDP is suitable for MANET applications due to low message overhead and comparable accuracy of diagnosis.

This work mainly focuses on fixed topology MANETs and the detection of permanent faults. Diagnosing MANETs in dynamic topology environment, and handling intermittent faults are more challenging tasks. These are perused in Chapter 4.

Chapter 4

Fault Diagnosis in Dynamic Topology MANETs

In contrast to the assumed static topology of the network in Chapter 3, dynamically changing network topology makes the diagnosis process more challenging. Permanently hard faults can be detected using the conventional *time-out* mechanism. Since permanently soft faults exhibit a continuous behavioral pattern of being faulty, they can easily be handled. However, the sporadic behavior of intermittently faulty nodes makes them difficult to diagnose. Elhadef *et al.* [32] have discussed a diagnosis protocol, called mobile-DSDP, to diagnose only permanently faulty nodes in time-varying topology MANETs. To the best of our knowledge, no fault diagnosis algorithm for MANETs in the literature has considered both permanent and intermittent faults in dynamic topology environment. This chapter presents a novel diagnosis protocol, called *flexible*-DSDP, that handles both the fault types, and unleashes the constraint on node's mobility.

The execution of a common test task, and comparison of the results forms the basis of diagnosis. Time-varying topology is handled by maintaining a preconfigured spanning tree, throughout the diagnosis session. The faulty nodes are to be excluded from the tree to follow a correct dissemination. Time redundancy is used to handle intermittent faults, by comparing the test outcomes for r rounds.

The rest of this chapter is organized as follows. The proposed *flexible*-DSDP for

dynamic topology MANET is discussed in Section 4.1. The correctness and completeness of the DSDP are analyzed in Section 4.2. The performance of the protocol is evaluated through simulation, and the obtained results are presented in Section 4.3. Finally, the work is summarized in Section 4.4.

4.1 Proposed *flexible*-DSDP

The proposed DSDP considers that a distributed ST that covers all hosts in the MANET is configured at the beginning, and is rooted at a robust node called the *initiator*. The tree is helpful for exchanging local and global diagnostics among the hosts. Mobile nodes, at the time of deployment, are assigned a common test task, say T. A node is required to execute T and respond with the result when it is asked by some neighbor, for the first time only.

The proposed protocol proceeds in four phases: *Maintenance* phase, *Comparison* phase, *Repairing* phase, and *Dissemination* phase. In the *Maintenance* phase, the pre-configured ST is maintained connected, considering the dynamically moving hosts. The *Comparison* phase focuses on classifying the hosts to be permanently faulty, intermittently faulty, or fault-free based on the comparison of results of execution of the common test task by them, for r rounds. The faulty parents are then removed from the ST in the *Repairing* phase. In the *Dissemination* phase, first the local views of the nodes are propagated to the root of the tree, which then generates the global diagnosis view and disseminates it down the tree to all fault-free hosts. These phases are discussed more vigilantly in the following subsections.

4.1.1 Maintenance Phase

As the hosts are considered to be mobile, a node may be disconnected from the tree and become *orphan* either because the parent is moving away from it or because the host itself is moving away from the parent. Therefore, it is required to maintain the ST connected. To do so, each mobile checks periodically if it is disconnected from the tree. In such case, the mobile seeks for a new parent by asking its neighbors to

adopt it. This is accomplished by sending a RECONNECT message to its neighbors. The steps followed by a mobile to be reconnected to the ST is given in Algorithm 4.1. The set of faulty nodes known to S_i is denoted as F_{S_i}, and is initialized to NULL. It has no impact in line 1 of Algorithm 4.1 during maintenance. However, it is relevant in *Repairing* phase of the diagnosis discussed in Section 4.1.3.

Algorithm 4.1: Reconnect To Spanning Tree

```
   Data: C(S, L^t): Communication graph of MANET.
         ST: Spanning Tree.
   Result: A disconnected mobile S_i reconnects to the ST.
   // ConnectedToST: Boolean variable initialized to TRUE; set to FALSE when the
   // mobile S_i finds that it is disconnected from the ST.
1  if Parent_{S_i} ∉ (N_{S_i} − F_{S_i}) then
2      if ConnectedToST then
3          ConnectedToST = FALSE;
4          _rb(RECONNECT, S_i);
5      end
6  end
```

To reduce the depth of the tree, mobiles respond with their depths after receiving messages to adopt an orphan. The orphan mobile then chooses the one having minimum depth, as parent. Moreover, even though a mobile is not disconnected from the tree, but observes that the *initiator* is one of its neighbors, it reconnects itself to the *initiator*, following Algorithm 4.2, to decrease the depth of the tree.

Algorithm 4.2: Connect To Initiator

```
   Data: C(S, L^t): Communication graph of MANET.
         ST: Spanning Tree.
   Result: Mobile S_i connects to initiator in the ST, if initiator ∈ N_{S_i}.
1  if initiator ∈ N_{S_i} and Parent_{S_i} ≠ initiator then
2      _rb(REMOVECHILD, S_i, Parent_{S_i});
3      Parent_{S_i} = initiator;
4      _rb(ADDCHILD, S_i, initiator, SentLD_{S_i}, ReceivedGD_{S_i});
5      ConnectedToST = TRUE;
6  end
```

Two types of messages are handled in this case:

(i) REMOVECHILD − Prior to choosing *initiator* as new parent, mobile S_i intimates its current parent about the same by sending a message of type REMOVECHILD.

(ii) AddChild − After adding *initiator* as its parent, mobile S_i propagates this event to the *initiator* by sending a message of type AddChild.

In order to reduce the communication complexity these two messages can be piggy-backed. Once reconnected, mobile S_i resets $ConnectedToST$ to $TRUE$. The implication of sending $SentLD_{S_i}$ and $ReceivedGD_{S_i}$ as a part of AddChild message will be discussed later in this section.

Every mobile S_i maintains its children set (denoted by $Children_{S_i}$) in the ST and updates it periodically. The update is necessary since the tree dynamically changes with moving hosts. During *Maintenance* phase (Algorithm 4.3), a mobile S_i receives and handles four types of messages:

(i) RemoveChild − Mobile S_i may receive RemoveChild message from $S_j \in Children_{S_i}$, if S_j wishes to be connected to the *initiator*. Upon receiving this message, S_i simply removes S_j from $Children_{S_i}$.

(ii) AddChild − Mobile S_i may receive AddChild message from S_j, *i.e.*, msg = (AddChild, $S_j, S_i, SentLD_{S_j}$), if S_j has already added it as parent. $SentLD_{S_j}$ is a boolean variable indicates if S_j has already sent its local diagnostic or not. It is initialized to FALSE and does not have any importance during *Maintenance* phase. However, it is useful during *Dissemination* phase of diagnosis. Upon receiving the AddChild message, mobile S_i adds S_j to its children list.

(iii) Reconnect − Mobile S_i may receive Reconnect message from S_j, *i.e.*, msg = (Reconnect, S_j), if S_j becomes orphan and wants to be reconnected to the ST. In this case, two scenarios may arise.

Scenario 1: $S_i \in Children_{S_j}$ − If S_j is the parent of S_i then it implies that S_i has also lost connection to the ST. In this case, after resetting $ConnectedToST$ to FALSE, mobile S_i seeks for a new parent in the tree by broadcasting a Reconnect message to its neighbors.

Algorithm 4.3: Maintenance Phase

Data: $C(S, L^t)$: Communication graph of MANET.
ST: Spanning Tree.
Result: ST is maintained connected.

```
1  msg = Message received from mobile S_j ∈ N_{S_i};
2  switch msg.Type() do
3      case RemoveChild: // i.e., msg = (RemoveChild, S_j, Parent_{S_j})
4          if S_i == Parent_{S_j} then
5              Children_{S_i} = Children_{S_i} − {S_j};
6          end
7      end
8      case AddChild: // i.e., msg = (AddChild, S_j, S_k, SentLD_{S_j}, ReceivedGD_{S_j})
9          if S_i == S_k and S_j ∉ F_{S_i} then
10             Children_{S_i} = Children_{S_i} ∪ {S_j};
11             if SentLD_{S_j} then
12                 ChildrenSentLD_{S_i} = ChildrenSentLD_{S_i} ∪ {S_j} ;
13             end
14             if !ReceivedGD_{S_j} and SentGD_{S_i} then
15                 _rb(m_{S_i} = (GlobalDiagnostic, GF_{S_i}, GF^P_{S_i}, GF^I_{S_i}));
16             end
17         end
18     end
19     case Reconnect: // i.e., msg = (Reconnect, S_j)
20         if ConnectedToST and S_j ∉ F_{S_i} then
21             if Parent_{S_i} == S_j then
22                 ConnectedToST = FALSE;
23                 _rb(Reconnect, S_i);
24             else
25                 _rb(AddParent, S_i);
26             end
27         end
28     end
29     case AddParent: // i.e., msg = (AddParent, S_j)
30         if !ConnectedToST and S_j ∉ F_{S_i} then
31             ConnectedToST = TRUE;
32             Parent_{S_i} = S_j;
33             _rb(AddChild, S_i, S_j, SentLD_{S_i}, ReceivedGD_{S_i});
34             _rb(AddParent, S_i);
35         end
36     end
37 end
```

Scenario 2: $S_i \notin Children_{S_v}$ − If S_i is connected to the ST and is not a child of S_j then it includes S_j in as its new child by transmitting a message of AddParent type.

(iv) AddParent − A mobile node S_i may receive many AddParent messages from its neighbors. However, if S_i is disconnected from the ST then it chooses the neighbor having lowest depth as its parent. Note that mobiles can forward their depth in the ST with the AddParent messages. After choosing a parent, say S_j, mobile S_i intimates this event by sending an AddChild message so that S_j can include it in $Children_{S_j}$. Along with this, S_i announces that it is now connected to the tree by sending an AddParent message to all its neighbors. These two messages can be piggybacked.

Note that the second condition, *i.e.*, $S_j \notin F_{S_i}$ in lines 9, 20, and 30 of Algorithm 4.3 does not have any significance during *Maintenance* phase. However, after faulty mobiles are detected, this condition is useful in *Repairing* phase to remove faulty nodes from the ST. This condition is included so that the same algorithm can be used in *Repairing* phase of the diagnosis session.

4.1.2 Comparison Phase

A mobile S_i starts its diagnosis session through *Comparison* phase either periodically, or if it receives a message of Result type that intimates it about the initiation of the diagnosis session by some other mobile. At this point, S_i sets a timer to T_{out} that helps in detecting the faulty stable neighbors those did not respond within the specified time. The *Comparison* phase is executed for r rounds to handle intermittently faulty mobiles.

During this phase, two types of messages are handled:

(i) Result − A mobile S_i may receive message of Result type from its neighbor S_j that has already started the diagnosis session, *i.e.*, S_i receives $msg = (\text{Result}, R_{S_j})$, where R_{S_j} is the result of execution of test task T by mobile S_j. After receiving such message, S_i starts the diagnosis session if it has not yet started, by broadcasting R_{S_i} through a message of Result type (line 7, Algorithm 4.4). Every mobile S_i maintains a list $Pending_{S_i}$, to store the results from the neighbors that are still not classified as correct or wrong. Two scenarios may arise at this point:

Algorithm 4.4: Comparison Phase

Data: $C(S, L^t)$: Communication graph of MANET.
ST: Spanning Tree.
Result: Each mobile S_i finds the local diagnosis view.
// mobile S_i executes this algorithm in each diagnostic round x

```
1  msg = Message received from mobile S_j ∈ N_{S_i};
2  switch msg.Type() do
3      case RESULT: // i.e., msg = (RESULT, R_{S_j})
4          if !DiagnosisStarted then
5              Activate timer T_out;
6              R_{S_i} = Compute task T;
7              _rb(m_{S_i} = (RESULT, R_{S_i}));
8              DiagnosisStarted = TRUE;
9          end
10         if ∃ < S_k, R_{S_k} > in Pendings_{S_i} | R_{S_j} == R_{S_k} then
11             FF^x_{S_i} = FF^x_{S_i} ∪ {S_j, S_k};
12             Pendings_{S_i} = Pendings_{S_i} − {< S_k, R_{S_k} >};
13         else
14             Pending_{S_i} = Pendings_{S_i} ∪ {< S_j, R_{S_j} >};
15         end
16     end
17     case TIMEOUT: // process pending list
18         ComparisonPhaseEnded = TRUE;
19         for each < S_k, R_{S_k} >∈ Pendings_{S_i} do
20             if R_{S_k} == R_{S_i} then
21                 FF^x_{S_i} = FF^x_{S_i} ∪ {S_k};
22             else
23                 F^x_{S_i} = F^x_{S_i} ∪ {S_k};
24             end
25             Pendings_{S_i} = Pendings_{S_i} − {< S_k, R_{S_i} >};
26         end
27     end
28 end
```

Scenairio 1: $\exists < S_k, R_{S_k} > in\ Pending_{S_i} | (R_{S_j} == R_{S_k}) -$ In this scenario, mobile S_i has already received the same result $R_{S_j} (= R_{S_k})$ from another mobile S_k. Therefore, following asymmetric comparison model, it can now classify both S_j and S_k as fault-free. Hence, S_i includes S_j and S_k in $FF^x_{S_i}$, where $FF^x_{S_i}$ denotes the set of fault-free mobiles detected by S_i in round x. The list $Pendings_{S_i}$ is updated accordingly to remove $< S_k, R_{S_k} >$ from it.

Scenairio 2: $\nexists < S_k, R_{S_k} > in\ Pending_{S_i} | (R_{S_j} == R_{S_k})$ − In this scenario, mobile S_i receives a new result R_{S_j} for which there is no matching existing result in $Pending_{S_i}$. At this point, S_i is not able to diagnose the status of S_j. So it simply adds the pair $< S_j, R_{S_j} >$ to $Pending_{S_i}$. Mobile S_i processes $Pending_{S_i}$ after T_{out} expires.

(ii) TimeOut − Once the timer T_{out} is fired, mobile S_i processes the list of pending results from mobiles for which it was unable to diagnose. For each $(S_k, R_{S_k}) \in Pending_{S_i}$, if R_{S_k} is equal to R_{S_i} then S_i classifies S_k as fault-free; otherwise as faulty. After processing, $< S_k, R_{S_k} >$ is removed from $Pending_{S_i}$.

At the end of r diagnostic rounds, each mobile finds the local diagnostic view constituting the fault status of the stable neighbors and of those dynamic neighbors from which it has received the Result messages.

4.1.3 Repairing Phase

During *Comparison* phase, each host S_i generates two sets $F_{S_i}^x$ and $FF_{S_i}^x$ in round x, where $F_{S_i}^x$ constitutes the set of faulty mobiles and $FF_{S_i}^x$ constitutes the set of fault-free mobiles detected by S_i in round x. Mobile S_i maintains two sets: F_{S_i} and FF_{S_i} as the local diagnostic view. Here, F_{S_i} is the set of $F_{S_i}^x$ and FF_{S_i} is the set of $FF_{S_i}^x$, for $1 \leq x \leq r$. The *Repairing* phase is useful to make sure that each mobile in the ST is connected with a fault-free parent. So, if a mobile S_i detects faulty parent, then it seeks for a new fault-free parent by invoking Algorithm 4.1, that in turn triggers Algorithm 4.3. The steps followed in this phase are similar to that of *Maintenance* phase. The only difference is that F_{S_i} may not be NULL in this case. Since faulty nodes are isolated from the ST in this phase, the *Maintenance* algorithm is termed as *Repairing* algorithm.

4.1.4 Dissemination Phase

The objective of the *Dissemination* phase is to generate global diagnostic view at each mobile by exchanging their local diagnostics. Recall that the local diagnostic

at mobile S_i is maintained by two sets F_{S_i} and FF_{S_i}. Once the ST is repaired and all faulty nodes are removed from it, all leaf nodes start disseminating their local diagnostics to their parents.

A mobile S_i has to handle two types of messages during this phase.

(i) LocalDiagnostic − After receiving such message from its child S_j, *i.e.* , msg = (LocalDiagnostic, F_{S_j}, FF_{S_j}), mobile S_i updates its local view to include the local view of S_j. Each mobile S_i maintains a set $ChildrenSentLD_{S_i}$, to keep track of the list of children those have already sent their local diagnostics. A mobile S_k in the set $ChildrenSentLD_{S_i}$ can be one of the following types:

- It has sent its local view to its old parent and now is a child of S_i. In this case, S_k is included in $ChildrenSentLD_{S_i}$ while maintaining the connectivity of the tree in Algorithm 4.3 (line 12).
- It has sent its local view to S_i but has been migrated to some other parent.
- It has sent its local view to S_i and still is a child of S_i.

Mobile S_i disseminates its local view only when all its current children have sent their local views.

If S_i is the initiator and has received local views of all other nodes in the tree, then it updates the faulty set in each round x to include those mobiles that have not been diagnosed by any other mobile. This scenario arises if a mobile moves out of the whole network. Now the initiator generates the global diagnostic view, classifying the mobiles to be permanently faulty, intermittently faulty, or fault-free.

(ii) GlobalDiagnostic − If a mobile S_i receives a GlobalDiagnostic message from its parent S_j for the first time, then it broadcasts the same to its neighbors. At this time it sets both $ReceivedGD_{S_i}$ and $SentGD_{S_i}$ to TRUE. Note that if a mobile S_i is disconnected from its parent during the *Dissemination* phase before receiving the global view from it, then it receives the same from its new parent. Moreover, by that time if the new parent has already sent its global view, then

Algorithm 4.5: Dissemination phase

Data: $C(S, L^t)$: Communication graph of MANET.
ST: Spanning Tree.
Result: Each mobile S_i has the global diagnostic view.

if $Children_{S_i} == \phi$ **then**
  $\_rb(m_{S_i} = (\text{LocalDiagnostic}, FF_{S_i}, F_{S_i}))$;
  $SentLD_{S_i} = TRUE$;
end
msg = Message received from mobile $S_j \in N_{S_i}$;
switch $msg.Type()$ **do**
  case LocalDiagnostic*: // i.e., msg* = $(\text{LocalDiagnostic}, F_{S_j}, FF_{S_j})$
    if $S_j \in Children_{S_i}$ **then**
      for $l = 1$ **to** r **do**
        $FF^l_{S_i} = FF^l_{S_i} \cup FF^l_{S_j}$;
        $F^l_{S_i} = F^l_{S_i} \cup F^l_{S_j}$;
      end
      $ChildrenSentLD_{S_i} = ChildrenSentLD_{S_i} \cup \{S_j\}$;
      if $(Children_{S_i} \cap ChildrenSentLD_{S_i}) == Children_{S_i}$ **then**
        $\_rb(m_{S_i} = (\text{LocalDiagnostic}, FF_{S_i}, F_{S_i}))$;
        $SentLD_{S_i} = TRUE$;
      end
      if $S_i == initiator$ **then**
        for $l = 1$ **to** r **do**
          $F^l_{S_i} = F^l_{S_i} \cup (S - (F^l_{S_i} \cup FF^l_{S_i}))$;
          // Compute Global Diagnostic View
          for each $S_k \in F^l_{S_i}$ **do**
            TimesFaulty[S_k]++;
          end
        end
        for each $S_k \in S$ **do**
          if $TimesFaulty[S_k] == r$ **then**
            $GF^P_{S_i} = GF^P_{S_i} \cup \{S_k\}$;
          else if $TimesFaulty[S_k] == 0$ **then**
            $GFF_{S_i} = GFF_{S_i} \cup \{s_k\}$;
          else
            $GF^I_{S_i} = GF^I_{S_i} \cup \{S_k\}$;
          end
        end
        $\_rb(m_{S_i} = (\text{GlobalDiagnostic}, GFF_{S_i}, GF^P_{S_i}, GF^I_{S_i}))$;
        $SentGD_{S_i} = TRUE$;
        $ReceivedGD_{S_i} = TRUE$;
      end
    end
  end
  case GlobalDiagnostic*:*
    if $S_j == Parent_{S_i}$ ***and*** $!ReceivedGD_{S_i}$ **then**
      $ReceivedGD_{S_i} = TRUE$;
      $\_rb(m_{S_i} = (\text{GlobalDiagnostic}, GFF_{S_i}, GF^P_{S_i}, GF^I_{S_i}))$;
      $SentGD_{S_i} = TRUE$;
    end
  end
end

it sends it again. This step is incorporated into *Maintenance* phase (line 14-16, Algorithm 4.3).

4.2 Analytical Study of proposed *flexible*-DSDP

This section presents the mathematical analysis to prove the correctness and completeness of the proposed protocol.

4.2.1 Correctness of *flexible*-DSDP

The *flexible*-DSDP is said to be correct, if no fault-free host $S_i \in S$ misdiagnoses the correct state of any other host in the MANET, at the end of the diagnosis session. The following lemmas are defined and proved to support the correctness of the protocol.

Lemma 4.1. *Let the undirected graph $C(S, L^t)$ represents the topology of the MANET, where S is the set of hosts, and L^t is the set of logical links among them at time t. Assume that a ST covering all hosts is maintained. If $S_i \in S$ is disconnected from ST then it is reconnected to the same during Maintenance phase.*

Proof. Once S_i finds that $Parent_{S_i}$ is no longer in its neighborhood, it sends a RECONNECT message (line 4, Algorithm 4.1) to all its neighbors. Neighbors except its own children confirm their wishes to adopt S_i by sending ADDPARENT messages (line 25, Algorithm 4.3). Upon receiving these messages, S_i adds the one having lowest depth in ST, say S_j, as parent and sends an ADDCHILD message to it (line 33, Algorithm 4.3). Host S_j then includes S_i in its children list (line 10, Algorithm 4.3). Thus node S_i is reconnected to the ST. □

Lemma 4.2. *Let S, the set of mobiles and L^t, the set of logical links at time t, collectively define an undirected graph $C(S, L^t)$, that represents the communication graph of the MANET. At the end of the Comparison phase for round x, the actual fault status of each host except permanent hard faulty ones, is known to at least one fault-free host.*

Proof. A fault-free host S_i, irrespective of whether it is dynamic or stable, initiates its diagnosis once it receives a message of RESULT type from any of its neighbors. At this point, it broadcasts a RESULT message after computing the common test task T. Upon receiving this message, its fault-free neighbor, say S_j, can deduce its correct status immediately if S_i has already received the same result from some other host and not processed yet; otherwise, the actual status can be deduced after time-out occurs (line 17, Algorithm 4.4). Same reasoning also applies when S_i is soft faulty dynamic or stable host. In this case, S_i sends the RESULT message with incorrect result and the fault-free neighbor S_j after receiving it can deduce the status after time-out. Note that the hard faulty hosts are detected after all local diagnostics have been sent to the initiator of the ST. The initiator then declares the undiagnosed hosts to be hard faulty (line 20, Algorithm 4.5). □

Lemma 4.3. *Let a MANET is represented by an undirected graph $C(S, L^t)$. The set of hosts in MANET is denoted as S and the set of logical links among them at time t is denoted as L^t. After Comparison phase, if a node is found to be connected to a faulty parent in the ST then it is reconnected to a fault-free parent during Repairing phase.*

Proof. If a node discovers its parent to be faulty then it triggers the *Repairing* phase by invoking Algorithm 4.1. The working of *Repairing* phase (Algorithm 4.3) is discussed previously. More specifically, the second condition, *i.e.*, $(S_j \notin F_{S_i})$ in lines 9, 20, and 30 of this algorithm helps S_i in choosing a fault-free parent. □

Lemma 4.4. *Let the topology of a MANET at an instance of time t is represented by an undirected graph $C(S, L^t)$, where S and L^t respectively represents the set of mobiles and the set of logical links at time t. The local diagnostic view of any fault-free host $S_i \in S$ reaches the initiator correctly during Dissemination phase.*

Proof. Any host S_i can verify if it is a leaf node in the ST. If $Children_{S_i}$ is NULL, then it starts disseminating its local view to its parent, say S_j. Node S_j keeps track of the children who has already sent the local diagnostic through a list $ChildrenSentLD_{S_j}$. Here, three scenarios need to be clarified. First, if S_i is a stable

neighbor of S_j then $S_i \in Children_{S_j}$, and S_j includes S_i in $ChildrenSentLD_{S_j}$. Second, if S_i has sent its local view to S_j but has been migrated to some other parent then $S_i \in ChildrenSentLD_{S_j}$ but $S_i \notin Children_{S_j}$. Third, if S_i has sent its local view to its old parent, but now has been migrated to S_j then $S_i \in ChildrenSentLD_{S_j}$ and $S_i \in Children_{S_j}$. This is due to the fact that if host S_i chooses a new parent S_j then it transmits a boolean field $SentLD_{S_i}$ in the AddChild message indicating if it has already sent its local view to the old parent or not. If it has done so then S_j includes S_i in $ChildrenSentLD_{S_j}$ (line 12, Algorithm 4.3). So $(Children_{S_j} \cap ChildrenSentLD_{S_j})$ represents the set of children of S_j who have already sent their local diagnostics. Host S_j collects local diagnostics from all its children and transmits a combined diagnostic message to its parent. This process continues and the initiator collects all the local diagnostic views. □

Theorem 4.1. (*Proof of Correctness*) *Let the set of hosts in the MANET and the set of logical links between them are represented by S and L^t respectively. The topology of MANET at time t is defined by an undirected graph $C(S, L^t)$. Every fault-free host in the ST correctly receives the global diagnostic information about the fault status of all other hosts in the MAENT.*

Proof. Lemma 1 and 3 conveys that the ST is always maintained connected and each node has a fault-free parent in the tree. In Lemma 2, it is proved that each host except the hard faulty ones is correctly diagnosed by at least one fault-free host in the MANET. The *initiator* receives the local diagnostic messages from all fault-free mobiles correctly. This is proved in Lemma 4. If the *initiator* finds a node to be faulty in all r rounds, then it is categorized to be permanently faulty and is added to the global permanent faulty set (line 28, Algorithm 4.5). Similarly, if a node is found to be faulty in no round, then it is added to the global fault-free set. However, if the node is found to be faulty in some rounds and fault-free in others, then it is added to the global intermittent faulty set. These three sets constitute the global diagnostic view. The *initiator* then transmits the global view to its children. Upon receiving the global view each node relays it to the nodes in the next lower level in the ST, and sets $ReceivedGD_{S_i}$ and $SentGD_{S_i}$ to TRUE to indicate that it has received and sent the

global diagnostic. This process continues until the global diagnostic message reaches the leaf nodes.

A special case must be clarified here. Consider a scenario where the parent of a node S_i, say S_j, has not yet disseminated the global view. If S_i finds that S_j is not in the communication range, then it seeks for a new parent, say S_k. Now, if S_k has already sent the global view, then S_i will not receive the global view. To handle this situation, the node S_i intimates S_k if it has already received the global view or not. This is accomplished by transmitting the boolean variable $ReceivedGD_{S_j}$ through AddChild message. Node S_k then retransmits the global view to S_i. Thus, each fault-free node receives the global diagnostic view correctly. □

4.2.2 Completeness of *flexible*-DSDP

The diagnosis is said to be complete, if no node in MANET is left undiagnosed by any of the fault-free hosts. The completeness of the *flexible*-DSDP is proved through the following theorem.

Theorem 4.2. (*Proof of Completeness*) *In an undirected graph $C(S, L^t)$, that represents the communication graph of MANET, where S and L^t subsequently represents the set of hosts and the set of logical links among them at time t, for any fault-free node S_i if $GF^P_{S_i}$, $GF^I_{S_i}$, and GFF_{S_i} are the set of permanently faulty, intermittently faulty, and fault-free hosts respectively then $(GF^P_{S_i} \cup GF^I_{S_i} \cup GFF_{S_i}) = S$.*

Proof. It can be followed from Lemma 2 that at the end of *Comparison* phase, the fault status (intermittently faulty, permanently soft faulty, or fault-free) of a node is known to at least one fault-free host in MANET. This information is correctly disseminated to the initiator as proved in Lemma 4. Moreover the permanently hard faulty nodes are detected by the initiator during the *Dissemination* phase. Hence, no node in MANET remains undiagnosed by the initiator. Thus the initiator deduces a complete global view which is correctly propagated to all fault-free nodes in the ST, as proved in Theorem 1. So $(GF^P_{S_i} \cup GF^I_{S_i} \cup GFF_{S_i}) = S$, at each fault-free node S_i in the MANET. □

Theorem 4.3. (*Communication Complexity*) *The communication complexity of the proposed protocol is $O(nd)$, where n is the number of nodes in the network and d is the average node degree.*

Proof. The message overhead of each phase of the algorithm can be found as follows.

(i) Maintenance phase − For each Reconnect message, d neighbors send AddParent message, and subsequently a AddChild message is sent from the disconnected node to the chosen parent. In the worst case, the number of Reconnect message is $d'(n-1)$, where d' is the average number of times a node is disconnected from the ST due to dynamic topology. This case arise when all nodes, except the initiator, have to send the Reconnect message.

Even though a node is not disconnected from the ST, it may wish to connect to the initiator. This case arises when a node finds the initiator as one of its neighbors, however, initiator is not its current parent. For each such case two message communications are required: one RemoveChild and one AddChild. If p is the average number of such cases then total number of message exchanges is $2p$.

So, total number of message communication in this phase is $d'(n-1)(d+2)+2p$ in worst case.

(ii) Comparison phase − In this phase, each node sends Result messages for r rounds. So, total number of Result messages is nr.

(iii) Repairing phase − In the worst case, each node except the initiator, executes the repairing phase by sending a Reconnect message. For each Reconnect message, d AddParent message and subsequently one AddChild message is sent. So maximum number of message exchanges in this phase is $(n-1)(d+2)$.

(iv) Dissemination phase − Each node, except the initiator, sends the LocalDiagnostic message. So, total number of LocalDiagnostic message exchanged is $n-1$. Each node, except the leaves, transmits GlobalDiagnostic message once. The maximum number of such message exchange is $n-1$, and this case

arises when the depth of the ST is $n-1$. So maximum number of message exchanges in this phase is $2(n-1)$.

Now, the message complexity of the protocol can be given as $O((d'+1)(n-1)(d+2)+nr+2p+2(n-1))$. Here, d', r, and p are invariant of n. So the communication complexity can be expressed as $O(nd)$. □

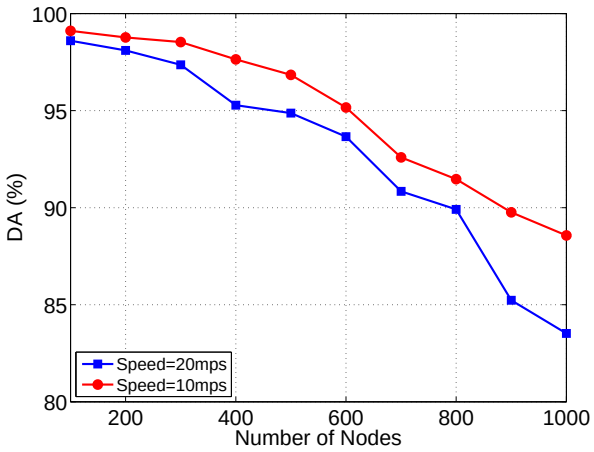
4.3 Simulation and Results

The performance of the proposed *flexible*-DSDP is the key discussion in this section. In order to characterize and study the behavior of the proposed protocol, a set of simulations has been executed using INET Framework over OMNet++ simulator. Keeping in mind the objective of fault diagnosis algorithm, two key performance measures, namely, Detection Accuracy (DA), and False Alarm Rate (FAR) are used to evaluate the performance of the proposed DSDP. Random waypoint mobility model is used to describe the non-static nature of the mobile hosts in MANET. For realizing the effect of mobility on the above said performance measures, two different speeds ($10mps$ and $20mps$) for the mobile hosts are considered. The pause time to define the random waypoint mobility is chosen to be $3s$. Following simulation scenarios are created to evaluate the performance of the proposed protocol.

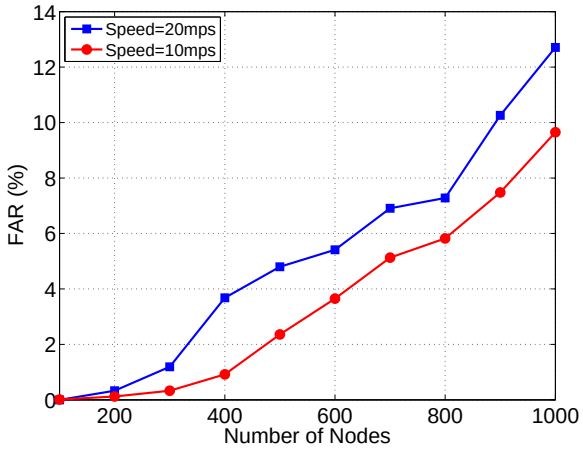
4.3.1 Simulation scenario 1

The first simulation scenario is created with 30% faulty nodes in the network. It is assumed that faulty nodes can be intermittently or permanently faulty with equal probability. The simulations are run for varied network size from 100 to 1000 mobiles, and the values for DA and FAR are recorded as shown in Figure 4.1. As expected, it is clear that the accuracy of detecting faulty nodes in the network decreases and accordingly FAR increases with the increase in network size. It can also be observed from Figure 4.1 that high mobility results in decreased DA and increased FAR.

The detection accuracy of the proposed DSDP is highly dependent on proper dissemination of the local diagnosis views of all the fault-free mobiles to the initiator.



(a)



(b)

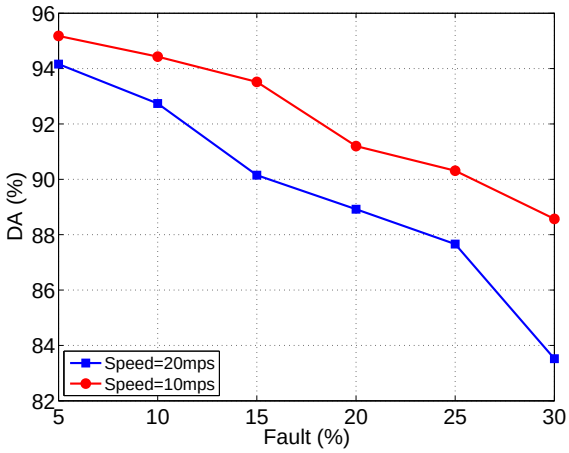
Figure 4.1: (a) DA, and (b) FAR, with varying network size.

However, due to the dynamic change in the topology of the MANET, the *Repairing* algorithm is executed very often which affects the dissemination process. The impact is even more in case of larger networks, and in case of highly dynamic environment. Another reason of loss of messages is due to interference of messages during communication, and the degree is high with increasing network size. This results in increasing FAR and decreasing DA.

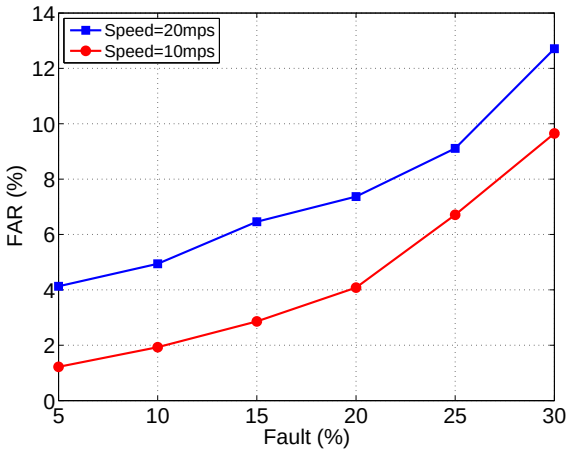
4.3.2 Simulation scenario 2

Another simulation scenario is set up with 1000 mobiles randomly deployed over a simulation area of 1000×1000 m^2 area. In this simulation the mobile nodes are

assumed to be faulty with probabilities 0.05, 0.10, 0.15, 0.20, 0.25, 0.30 respectively. The obtained DA and FAR values are shown in Figure 4.2. It conveys that with the increase in fault probability, the accuracy decreases and the miss detection rate increases.



(a)



(b)

Figure 4.2: (a) DA, and (b) FAR, with varying fault %.

This is expected because, for any node to be diagnosed properly, it requires at least one fault-free neighbor. However, with the increase in number of faulty nodes in the network, the probability of the communication graph (with fault-free nodes) to be disconnected increases, which in turn affects the correct diagnosis; resulting higher FAR and lower DA. Furthermore, more is the speed of the mobiles more is the chance of getting the ST disconnected. This affects the dissemination of local diagnostics to

the initiator, though the ST is repaired during *Repairing* phase. This further affects the DA and FAR.

4.4 Summary

In this Chapter, a novel distributed fault diagnosis protocol, called *flexible*-DSDP, is discussed that can handle both permanent and intermittent faults in MANETs. None of the literature considers handling intermittent faults with time-varying topology of the MANETs during diagnosis, whereas the proposed *flexible*-DSDP is designed to cope with dynamically changing topology of the network. The correctness and the completeness of the protocol are advocated by analytical proofs. The effectiveness of the proposed DSDP is supplemented with the obtained simulation results for an acceptable accuracy and false alarm rate, even in case of larger networks and with high fault probability. It is also observed that the protocol results in decreasing DA and increasing FAR if the network topology is highly dynamic. The protocol assumes that no new fault arises during diagnosis. Dynamic faults are more intractable compared to static faults. In future, an endeavor will be made examine the effect of such faults on accuracy of diagnosis.

Chapter 5

Fault Diagnosis in Static Topology WSNs

The inherent characteristics of WSNs, and the nature of critical applications demand for a highly reliable fault diagnosis algorithm for such networks. Based on the spatial correlation of sensor measurements, many fault diagnosis protocols have been proposed in the literature. The spatial correlation property states that the neighboring fault-free sensor nodes read similar measurements at any point of time, and the difference between the readings does not exceed a given threshold. To take a more accurate decision about the fault status of the sensor nodes, the remaining energy levels of the neighboring sensor nodes can also be compared. This exploits the fact that in homogeneous sensor networks, where the nodes perform similar tasks, the residual energy levels of fault-free neighbors do not differ significantly.

This chapter presents a fault diagnosis algorithm for WSNs that handles permanent as well as intermittent faults. To handle the intermittent faults, the comparisons are made for r rounds. Two special cases of intermittent faults are considered: one, where an intermittently faulty node sends similar sensor measurement and similar residual energy to some of its neighbors in all r rounds; another, where at least one of these quantities differ significantly from that of some of its neighbors in all r rounds.

The rest of this chapter is organized as follows. The proposed fault diagnosis algorithm is discussed in Section 5.1. Section 5.2 demonstrates the analytical study

to prove the correctness and completeness of the proposed algorithm. The simulation results are shown in Section 5.3 that vows the effectiveness of the protocol. A summary of the work is given in Section 5.4.

5.1 Proposed Fault Diagnosis Algorithm

The proposed fault diagnosis algorithm (FDA) eyes on the detection of nodes with following fault types:

- permanent, and intermittent faults in sensors
- permanent, and intermittent faults in communication units

Sensor nodes with permanently faulty communication unit can be detected with conventional *time-out* mechanism, and are to be excluded from the network. However, the nodes with malfunctioning sensors still remain associated with the network since they have the ability to relay data packets among the nodes. Therefore, our main focus is to detect the permanent faults in sensors and intermittent faults in sensors or communication units.

A *test graph*, $C' = (S', L')$, can be constructed from the communication graph, $C = (S, L)$, by excluding the nodes with permanently faulty communication units and the links associated with those nodes. So $S' \subseteq S$, $L' \subseteq L$, and C' is a subgraph of C. The number of faulty neighbors for any node $S_i \in S'$ is upper bound by $(\lceil |N_{S_i}|/2 \rceil - 1)$. This is an implicit requirement for a conventional majority voting protocol [60, 96, 97] which is followed in the proposed algorithm. In addition, it is assumed that each sensor node has the knowledge of the identity of its 1-hop neighbors along with their neighborhood information. This is a feasible requirement in sensor networks [98, 99, 100].

The proposed algorithm executes in two phases. In the first phase (*Comparison phase*), each fault-free node compares its sensor reading and residual energy level with that of the neighboring sensor nodes to classify them as fault-free, intermittently faulty, or permanently faulty.

In *Dissemination phase*, local diagnosis views are exchanged to generate the global view at each fault-free node, using a ST overlaying all fault-free nodes in the network. These phases are discussed in sequel.

5.1.1 Comparison Phase

Considering the spatial correlation in sensor networks, the measurement difference of two fault-free neighboring sensors is presumed to be very small. However, if at least one of them is faulty then the difference is significant. Hence, if $x^t_{S_i}$ is the sensor measurement of node S_i at a given time t, and $l_{(S_i,S_j)} \in L'$ then

$$|x^t_{S_i} - x^t_{S_j}| \begin{cases} \leq \delta_1, & \text{if both } S_i \text{ and } S_j \text{ are fault-free} \\ > \delta_1, & \text{otherwise.} \end{cases} \tag{5.1}$$

To aid the diagnosis process the residual energy levels of neighboring sensor nodes are also compared. In a homogeneous sensor network, the nodes in close proximity have similar levels of residual energy since they do the same duty [59, 101]. Hence, if $E^t_{S_i}$ represents the residual energy of node S_i and $S_j \in N^t_{S_i}$ then

$$|E^t_{S_i} - E^t_{S_j}| \begin{cases} \leq \delta_2, & \text{if both } S_i \text{ and } S_j \text{ are fault-free} \\ > \delta_2, & \text{otherwise.} \end{cases} \tag{5.2}$$

In Equations 5.1 and 5.2, δ_1 and δ_2 are two predefined thresholds. These thresholds may vary depending on the application.

Now, the comparison outcome, $c^t_{(S_i,S_j)}$, of any two neighboring nodes S_i and S_j can be defined as follows

$$c^t_{(S_i,S_j)} = \begin{cases} 0, & \text{if } |x^t_{S_i} - x^t_{S_j}| \leq \delta_1 \text{ and } |E^t_{S_i} - E^t_{S_j}| \leq \delta_2 \\ 1, & \text{otherwise.} \end{cases} \tag{5.3}$$

In Equation 5.3, $c^t_{(S_i,S_j)} = 0$ signifies both S_i and S_j are fault-free. But, if at least one of S_i and S_j is faulty then $c^t_{(S_i,S_j)} = 1$.

Each sensor node, $S_i \in S$ maintains a boolean status array $StatR_{S_i}[]$ of size n, to

store the fault status of all the nodes in the network. Initially, all 1-hop neighbors are assumed to be fault-free (0) and the status of all non-neighoring nodes are unknown (-1) as given in Equation 5.4.

$$StatR_{S_i}[j] = \begin{cases} -1, & \text{if } S_j \notin N_{S_i} \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

In each round, upto total of r rounds, each sensor node $S_j \in S'$ sends message of type DATA, containing its observed sensor reading and residual energy level, to $S_i \in N_{S_j}$, *i.e.*, $m_{S_j} = (\text{DATA}, x^t_{S_j}, E^t_{S_j})$. Upon receiving such m_{S_j} from its neighbor S_j, node S_i performs the threshold test defined in Equation 5.3 and increments $StatR_{S_i}[j]$ by 1, if at least one of the test conditions fails. At the end of r rounds, each sensor finds a local diagnosis view about the 1-hop neighbors. However, these local views may not be correct. Consider the following cases:

Case-I

Node S_i may misdiagnose an intermittently faulty neighbor S_j as fault-free. This case arises if S_j sends similar sensor measurement and similar residual energy value to S_i in all rounds.

To overcome this situation, S_i follows majority voting among the decisions of all $S_k \in N_{S_j}$ about the fault status of S_j. However, the views of all nodes $S_k \in (N_{S_i} \cap N_{S_j})$ that are already detected as faulty by S_i can be discarded to reduce the computation overhead. We consider the maximum number of neighbors to which S_j may send such similar values in all rounds is $\lceil n^+_{S_j}/2 \rceil - 1$, where $n^+_{S_j}$ represents the number of fault-free neighbors of S_j. Of course, this necessitates that each node sends the local view to its 2-hop neighbors.

So S_i finds the set of neighbors of S_j that are not yet detected faulty by S_i. Let this set be denoted as $NYF_{(S_i,S_j)}$ and defined as in Equation 5.5.

$$NYF_{(S_i,S_j)} = \{S_k | S_k \in N_{S_j} \text{ and } StatR_{S_i}[k] \leq 0\}. \tag{5.5}$$

Let $NYF0_{(S_i,S_j)} \subseteq NYF_{(S_i,S_j)}$ represents the set of neighbors of S_j in $NYF_{(S_i,S_j)}$ that

Algorithm 5.1: Comparison Phase

Data: $T = (S', L')$: The test graph.
r: number of rounds.
Result: $StatR_{S_i}[]$ for each fault-free node $S_i \in S'$
Initialization: NYF=0; NYF0=0; StatSum=0;

1 **for** ***each*** $S_i \in S'$ ***and*** $S_j \in S'$ **do**
2 **if** $S_i == S_j$ ***or*** $l_{(S_i,S_j)} \in L'$ **then**
3 $StatR_{S_i}[j] = 0$;
4 **else**
5 $StatR_{S_i}[j] = -1$;
6 **end**
7 **end**
8 **for** *r rounds* **do**
9 **for** ***each*** $S_j \in S'$ ***and*** $S_i \in N_{S_j}$ **do**
10 S_j sends a message $m_{S_j} = (\text{DATA}, x^t_{S_j}, E^t_{S_j})$;
11 **end**
12 **if** *a node* S_i *receives a message* m_{S_j} *from* $S_j \in N_{S_i}$ **then**
13 **if** $|x^t_{S_i} - x^t_{S_j}| > \delta_1$ ***or*** $|E^t_{S_i} - E^t_{S_j}| > \delta_2$ **then**
14 Increment $StatR_{S_i}[j]$;
15 **end**
16 **end**
17 **end**
18 Each node sends status array to its 1-hop and 2-hop neighbors;
19 **for** ***each*** $S_i \in S'$ ***and*** $S_j \in N_{S_i}$ **do**
20 **for** ***each*** $S_k \in N_{S_j}$ **do**
21 **if** $StatR_{S_i}[k] \leq 0$ **then**
22 Increment NYF;
23 **if** $StatR_{S_k}[j] == 0$ **then**
24 Increment $NYF0$;
25 **end**
26 **end**
27 **end**
28 **if** $NYF0 \geq \lceil NYF/2 \rceil$ **then**
29 $StatR_{S_i}[j] = 0$;
30 **else if** $StatR_{S_i}[j] = 0$ **then**
31 $StatR_{S_i}[j] = 1$;
32 **end**
33 **if** $StatR_{S_i}[j] > 0$ **then**
34 **for** ***each*** $S_k \in N_{S_j}$ **do**
35 $StatSum = StatSum + StatR_{S_k}[j]$;
36 **end**
37 **if** $StatSum == (r \times |N_{S_j}|)$ **then**
38 $StatR_{S_i}[j] = 2$; // $StatR_{S_i}[j] = 2$ indicates S_j is permanently faulty.
39 **else**
40 $StatR_{S_i}[j] = 1$; // $StatR_{S_i}[j] = 1$ indicates S_j is intermittently faulty.
41 **end**
42 **end**
43 **end**

have detected S_j to be fault-free, *i.e.*,

$$NYF0_{(S_i,S_j)} = \{S_k | S_k \in NYF_{(S_i,S_j)} \text{ and } StatR_{S_k}[j] = 0\}. \tag{5.6}$$

Now S_i can follow majority voting, and update its status array as follows:

$$StatR_{S_i}[j] = \begin{cases} 0, & \text{if } |NYF0_{(S_i,S_j)}| \geq \left\lceil \frac{1}{2}|NYF_{(S_i,S_j)}| \right\rceil \text{ and } StatR_{S_i}[j] \neq 0 \\ 1, & \text{if } |NYF0_{(S_i,S_j)}| < \left\lceil \frac{1}{2}|NYF_{(S_i,S_j)}| \right\rceil \text{ and } StatR_{S_i}[j] = 0. \end{cases} \tag{5.7}$$

In Equation 5.7, the value 0 or 1 of $StatR_{S_i}[j]$ indicates S_j to be fault-free or intermittently faulty respectively. Nevertheless, we can consider any positive value less than r to signify intermittent fault. Additionally, majority voting can cope with packet losses to a certain extent. For instance, loss of packets in some rounds from a fault-free neighbor, which affects the decision of S_i can be overridden.

Case-II

Node S_i may misdiagnose an intermittently faulty neighbor S_j as permanently faulty. This case arises if S_j sends sensor measurement and residual energy value, either or both of which deviates significantly from that of S_i in all rounds.

To handle this situation and to determine the actual fault type, S_i follows Equation 5.8, if $StatR_{S_i}[j] > 0$. The equation is based on the rationale that a permanently faulty node is always detected as faulty by all neighbors in all rounds.

$$StatR_{S_i}[j] = \begin{cases} 1, & \text{if } \left(\sum_{(S_k \in N_{S_j})} StatR_{S_k}[j]\right) \neq r \times |N_{S_j}| \\ 2, & \text{otherwise.} \end{cases} \tag{5.8}$$

The values 1 or 2 of $StatR_{S_i}[j]$ in Equation 5.8 signifies S_j to be intermittently faulty or permanently faulty respectively. The steps followed in this phase are more precisely described in Algorithm 5.1.

5.1.2 Dissemination Phase

The objective of the *Dissemination* phase is to construct a ST covering all fault-free nodes in the network, and generate global diagnostic view at each fault-free node by exchanging their local diagnostics over the ST. The operation of this phase is similar to the *Dissemination* phase discussed in Chapter 3 (Section 3.1.2). However, the diagnostic view of S_i, *i.e.*, $VIEW_{S_i}$ is composed of $StatR_{S_i}[]$.

5.2 Analytical Study of Proposed FDA

In this section we follow mathematical analysis to prove the correctness and completeness of the proposed algorithm.

5.2.1 Proposed FDA Correctness

If every fault-free sensor $S_i \in S'$ deduces the correct state of all other sensors in the network, then the proposed FDA is said to be correct. This can be defined with respect to *correct local diagnosis* and *correct dissemination* of local and global diagnosis information. We define the following lemmas to support the correctness of the proposed FDA.

Lemma 5.1. *Let $C' = (S', L')$ be the test graph for the sensor network, where S' and L' respectively represents the set of sensor nodes without permanently faulty communication unit and the set of logical links between them. If $S_j \in S'$ is an intermittently faulty node, then at the end of Comparison phase, it is neither misdiagnosed as fault-free nor as permanently faulty by any other sensor node $S_i \in (N_{S_j} \cap FF)$; where $FF \subseteq S'$ is the set of fault-free nodes.*

Proof. In case, an intermittently faulty sensor node $S_j \in S'$ sends to $S_i \in (N_{S_j} \cap FF)$, similar sensor measurement and similar residual energy value in all rounds; S_i misdiagnoses S_j as fault-free. However, this misdiagnosis is resolved once S_i follows the majority voting defined in Equation 5.7, after receiving the local views of all neighbors of S_j. The validation of Equation 5.7 can be given as follows.

If S_i finds any node $S_k \in N_{S_j}$ to be faulty then it discards the decision of S_k from voting. Hence, the maximum number of neighbors of S_j that may take part in voting is $|N_{S_j}|$. This case arises when none of the neighbors of S_j have been detected faulty by S_i. The minimum number of neighbors of S_j that may take part in voting

is $\lfloor |N_{S_j}|/2 \rfloor + 1$. This case arises when S_j has maximum number of faulty neighbors, *i.e.*, $\lceil |N_{S_j}|/2 \rceil - 1$, and all of them are in the neighborhood of S_i and are correctly detected faulty by S_i. Hence,

$$\lfloor |N_{S_j}|/2 \rfloor + 1 \leq |NYF_{(S_i,S_j)}| \leq |N_{S_j}|. \tag{5.9}$$

The maximum number of fault-free neighbors that may detect S_j as fault-free is $\lceil (\textit{minimum number of fault-free neighbors of } S_j)/2 \rceil - 1$, *i.e.*,

$$|NYF0_{(S_i,S_j)}| \leq \left\lceil \frac{\lfloor |N_{S_j}|/2 \rfloor + 1}{2} \right\rceil - 1. \tag{5.10}$$

Now consider a worst case scenario where $|NYF_{(S_i,S_j)}|$ is the minimum, *i.e.*, $\lfloor |N_{S_j}|/2 \rfloor + 1$ and $|NYF0_{(S_i,S_j)}|$ is the maximum, *i.e.*, $\left\lceil \frac{\lfloor |N_{S_j}|/2 \rfloor + 1}{2} \right\rceil - 1$. The condition $|NYF0_{(S_i,S_j)}| < \left\lceil \frac{1}{2}|NYF_{(S_i,S_j)}| \right\rceil$ holds. Hence, if S_i has misdiagnosed S_j as fault-free, *i.e.*, $StatR_{S_i}[j] = 0$, then it would update $StatR_{S_i}[j]$ to 1 (line 31, Algorithm 5.1); thus classifying S_j correctly as intermittently faulty.

It is also necessary to prove that S_i, after correctly diagnosing the intermittently faulty neighbor S_j, does not override its decision by subsequently following Equation 5.8. This can be verified with the following reasoning.

The inequality

$$\left(\sum_{(S_k \in N_{S_j})} StatR_{S_k}[j] \right) \neq r \times |N_{S_j}|$$

holds, since there is a non zero number of fault-free neighbors of S_j that have correctly diagnosed S_j, *i.e.*, for some fault-free node $S_k \in N_{S_j}$, $0 < StatR_{S_k}[j] < r$. Therefore, $StatR_{S_i}[j]$ is not changed to permanently faulty. Hence, each node $S_i \in (N_{S_j} \cap FF)$ correctly diagnoses the state of an intermittently faulty neighbor S_j at the end of the *Comparison* phase. □

Lemma 5.2. *Let S', the set of sensor nodes without permanently faulty communication unit and L', the set of logical links between them, collectively define an undirected graph $C' = (S', L')$ that represents the test graph of a sensor network. Let $FF \subseteq S'$ be the set of fault-free nodes in the network. At the end of the Comparison phase, the status of a fault-free node S_j is correctly detected by each node $S_i \in (N_{S_j} \cap FF)$.*

Proof. It is clear from Equations 5.1 and 5.2 that a fault-free node $S_j \in S'$ can be misclassified as faulty by a node $S_i \in \{N_{S_j} \cap FF\}$, if S_j sends to S_i: its sensor measurement and residual energy value, either or both of which deviates significantly from

that of S_i in at least one round. Nevertheless, by definition, a fault-free node always sends similar sensor measurement and never gives significantly different remaining energy value compared to its neighbors. So S_j can never fail any of the threshold tests performed by S_i in Equations 5.1 and 5.2, in any round. Hence, it is properly classified as fault-free by all $S_i \in (N_{S_j} \cap FF)$. Moreover, this decision is not affected by the majority voting. The reasoning is as follows.

For each node $S_i \in (N_{S_j} \cap FF)$, the inequality in Equation 5.9 holds and the explanation is same as given in Lemma 5.2. The minimum number of neighbors, out of those participated in voting, that have correctly identified S_j as fault-free can be given as in Equation 5.11.

$$|NYF0_{(S_i,S_j)}| \geq \left(\lfloor |N_{S_j}|/2 \rfloor + 1\right). \tag{5.11}$$

From Equations 5.9 and 5.11, it can easily be verified that,

$$|NYF0_{(S_i,S_j)}| \not< \left\lceil \frac{1}{2}|NYF_{(S_i,S_j)}| \right\rceil. \tag{5.12}$$

So the majority voting in Equation 5.7 would not affect the decision of S_i about its fault-free neighbor S_j. Furthermore, Equation 5.8 is followed by S_i if $StatR_{S_i}[j] > 0$. So, it cannot alter the value of $StatR_{S_i}[j]$ in this case. Therefore, a fault-free node S_j is always diagnosed as fault-free by each node $S_i \in (N_{S_j} \cap FF)$ at the end of the *Comparison* phase. □

Lemma 5.3. *Let the test graph of a sensor network be represented as $C' = (S', L')$; where S' is the set of sensor nodes having fault-free communication unit and L' is the set of logical links between them. Let $FF \subseteq S'$ represents the set of fault-free nodes. If a node S_j is having permanently faulty sensor, then it is neither diagnosed as fault-free nor as intermittently faulty by any node $S_i \in (N_{S_j} \cap FF)$.*

Proof. By convention, a node with permanently malfunctioning sensor (say S_j) gives wrong sensor measurements to all its neighbors, in all rounds. A node $S_i \in \{N_{S_j} \cap FF\}$ increments $StatR_{S_i}[j]$ by one, upon receiving wrong sensor measurements from S_j, in each round. So at the end of r rounds, $StatR_{S_i}[j]$ carries the value r. It can easily be seen, as follows, that the majority voting does not alter this decision. The inequality in Equation 5.9 holds due the same reasoning as described in Lemma 5.1, and $|NYF0_{(S_i,S_j)}| = 0$. Hence, it is clear that

$$|NYF0_{(S_i,S_j)}| < \left\lceil \frac{1}{2}|NYF_{(S_i,S_j)}| \right\rceil, \tag{5.13}$$

since all fault-free neighbors of S_j have properly identified it and in no round, two

faulty nodes generate the same incorrect measurements. However, Equation 5.7 does not alter $StatR_{S_i}[j]$ since $StatR_{S_i}[j] \neq 0$, that is because S_j is permanently faulty.

It is also important to validate Equation 5.8 for this scenario. For permanently faulty node S_j,

$$\left(\sum_{(S_k \in N_{S_j})} StatR_{S_k}[j] \right) = r \times |N_{S_j}|$$

holds, due to the fact that each $S_k \in N_{S_j}$, irrespective of whether it is faulty or not, has $StatR_{S_k}[j] = r$ at the end of diagnosis. So, $StatR_{S_i}[j]$ is set to a value 2, signifying it as permanently faulty. Hence, it can be concluded that S_j; a node with permanently faulty sensor is always diagnosed correctly by any node $S_i \in (N_{S_j} \cap FF)$. □

Lemma 5.4. (*Correct Local Diagnosis View*). *In an undirected graph, $C' = (S', L')$ that represents the test graph of a sensor network, where S' is the set of sensor nodes without permanently faulty communication unit and L' is the set of logical links between them, let $FF \subseteq S'$ represents the set of fault-free in the network. If a node S_j is fault-free, it is never misdiagnosed as faulty, and if it is faulty with certain fault type (permanent or intermittent), then it is neither misclassified as fault-free nor as a wrong fault type by any other node $S_i \in (N_{S_j} \cap FF)$.*

Proof. The proof of this lemma directly follows from Lemma 5.1, 5.2, and 5.3. Referring Lemma 5.2, if a node S_j is fault-free, then it is always diagnosed as fault-free by any fault-free node in N_{S_j}. Lemma 5.1 asserts that, if S_j is an intermittently faulty node, then it is never classified as fault-free, nor as permanently faulty by any other node $S_i \in (N_{S_j} \cap FF)$. Again, the permanently faulty nodes are always detected as permanently faulty by all fault-free neighbors, that is vouched by Lemma 5.3. □

Lemma 5.5. *Let the test graph of a sensor network be represented as $C' = (S', L')$; where S' is the set of sensor nodes having fault-free communication unit and L' is the set of logical links among them. The ST constructed during Dissemination phase contains all and only fault-free nodes in the network.*

Proof. The proof of this lemma follows Lemma 3.4. □

Lemma 5.6. (*Correct dissemination*) *Let the undirected graph, $C' = (S', L')$ represents the test graph of a sensor network, where S' is the set of sensor nodes without permanently faulty communication unit and L' is the set of logical links between them. At the end of the Dissemination phase, each node in the ST has the global view of the network.*

Proof. The proof of this lemma follows Lemma 3.5. □

Theorem 5.1. (*Proposed FDA Correctness*) *In an undirected graph, $C' = (S', L')$ that represents the test graph of a sensor network, where S' is the set of sensor nodes without permanently faulty communication unit and L' is the set of logical links between them, let $FF \subseteq S'$ represents the set of fault-free in the network. At the end of the diagnosis, each node $S_i \in FF$ has the consistent global diagnostic view of the network.*

Proof. The proof of this theorem follows Lemma 5.4, 5.5, and 5.6. Lemma 5.4 asserts that the local diagnosis view of any node $S_i \in FF$ is correct. According to Lemma 5.6, these local views are properly disseminated to the root of the ST. Subsequently, the global view generated at the root is disseminated to all nodes in the ST. Moreover, the ST overlays all and only fault-free nodes which is proved in Lemma 5.5. Therefore, each node $S_i \in FF$ has the consistent global diagnostic view of the network, at the end of the diagnosis session. □

5.2.2 Proposed FDA Completeness

The proposed diagnosis algorithm is said to be complete if no node in the network is left undiagnosed, and the proof can be given as in Theorem 2.

Theorem 5.2. (*Proof of Completeness*) *Let the test graph of a sensor network is represented as an undirected graph $C' = (S', L')$, where S' is the set of sensor nodes with good communication unit and L' is the set of logical links between them. If a node $S_i \in S'$, then at the end of the fault diagnosis session, $S_i \in (FF \cup F_P \cup F_I)$, i.e., $S' = (FF \cup F_P \cup F_I)$, where $FF \subseteq S'$, $F_P \subset S'$, and $F_I \subset S'$ respectively represent the set of fault-free, permanently faulty, and intermittently faulty nodes in the network.*

Proof. From Lemma 5.1 and 5.3 it is clear that, if a node S_i is faulty, then at the end of the fault diagnosis session, $S_i \in F_P$ *(if S_i is permanently faulty)* or $S_i \in F_I$ *(if S_i is intermittently faulty).* Lemma 5.2 conveys that, if S_i is fault-free then $S_i \in FF$, at the end of the fault diagnosis session. Hence, it can be concluded that, if $S_i \in S$ then $S_i \in (FF \cup F_P \cup F_I)$; thus $S' = (FF \cup F_P \cup F_I)$ at the end of the fault diagnosis session. □

Theorem 5.3. *The communication overhead of the proposed FDA is $O(nd)$, where n is the number of nodes in the network and d is the average node degree.*

Proof. In the worst case scenario, all the nodes in the network are fault-free. The message complexity of each phase of the algorithm can be found as follows.

(i) Comparison Phase − Each node sends its sensor measurement and residual energy value for r rounds. So the total number of DATA messages is nr. Each node broadcasts it local view to its 1-hop and 2-hop neighbors. Total number of such broadcasts is $n(d+1)$. This stems from the fact that each node broadcasts its local view, and subsequently all 1-hop neighbors broadcast the received local view to make it available at 2-hop neighbors of the sender.

(ii) Building Phase − Each node sends SPANTREE message once. Hence, total number of messages exchanged during the construction of the ST is n.

(iii) Dissemination Phase − Each node, except the *initiator*, sends the LOCALDIAGNOSTIC message. So, total number of LOCALDIAGNOSTIC message exchanged is $n-1$. Each node, except the leaves, sends transmits GLOBALDIAGNOSTIC message once. The maximum number of such message exchange is $n-1$, and this case arises when the depth of the ST is $n-1$.

Thus, the total number of messages exchanged during diagnosis is $n(r+d+4)-2$, and the communication complexity is $O(nd)$, since r is invariant of n. □

5.3 Simulation and Results

The performance evaluation of the proposed FDA is presented in this section. A set of simulations is performed using the Castalia-3.2 simulator on OMNeT++ 4.2 platform, to study the feasibility of the proposed algorithm. The results are compared with that of the detection algorithm discussed by Lee and Choi [62]. Their algorithm follows a round based approach to diagnose the nodes. A faulty node that generates incorrect sensor measurement in all rounds is considered permanently faulty, otherwise it is categorized as intermittently faulty. To have more accurate decision about the fault status of a node, voting is followed among the decisions of all neighbors regarding the same. A similar (round based with thresholding) approach is followed in our proposed algorithm. Hence, these two algorithms are comparable. However, Lee and Choi's algorithm uses a fixed threshold (θ_1) for voting that is not promising and may affect the accuracy. For instance, if $\theta_1 = 3$ and a fault-free node has two neighbors, then the node is not detected as fault-free, even in case both the neighbors are fault-free. In contrast, an adaptive threshold for each node would be more appropriate and can

improve the accuracy. The proposed algorithm employs such a threshold through majority voting technique.

Based on the faulty behavior, the proposed FDA classifies the sensor nodes into three different classes: permanent fault class, intermittent fault class, and fault-free class. The following two performance measures are used as the evaluation metrics.

- Classification Accuracy (CA): The ratio of the number of nodes classified into a particular class to the total number of nodes of that class. Hence, CA is defined with respect to each of the three classes mentioned above.

- False Alarm Rate (FAR): The ratio of the sum of the number of faulty nodes classified as fault-free and the number of fault-free nodes classified as faulty to the total number of nodes in the network.

FAR is defined with an observation that the severity of a fault-free node being misdiagnosed as faulty or a faulty node as fault-free is more, as compared to a permanently faulty node being misdiagnosed as intermittently faulty or vice versa. The performance of the proposed algorithm depends on many factors, *viz.*,

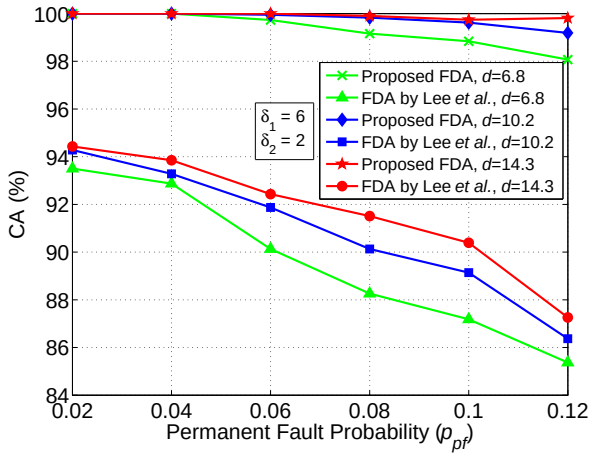
- $d-$the average node degree.

- $p_{pf}-$the probability that a sensor node has a permanently faulty sensor.

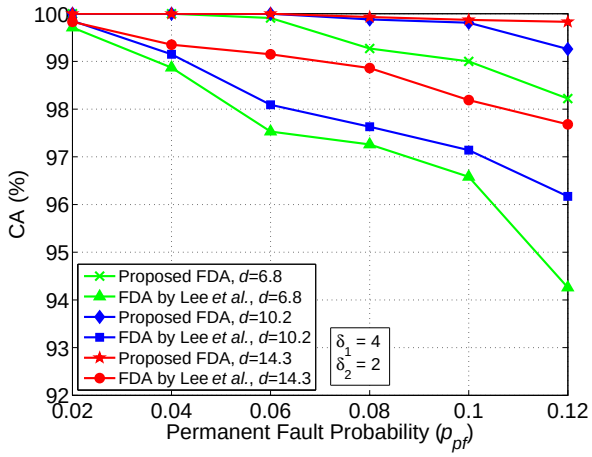
- $p_{if}-$the probability that a sensor node has an intermittently faulty sensor.

- $p_{fc}-$the probability that a sensor node has a faulty communication unit.

- δ_1, δ_2-thresholds used for sensor measurement deviation and residual energy deviation of healthy neighboring nodes.



(a)



(b)

Figure 5.1: Comparison of permanent fault CA with (a) δ_1=6, δ_2=2 and (b) δ_1=4, δ_2=2.

The following two simulation scenarios are considered for discussion.

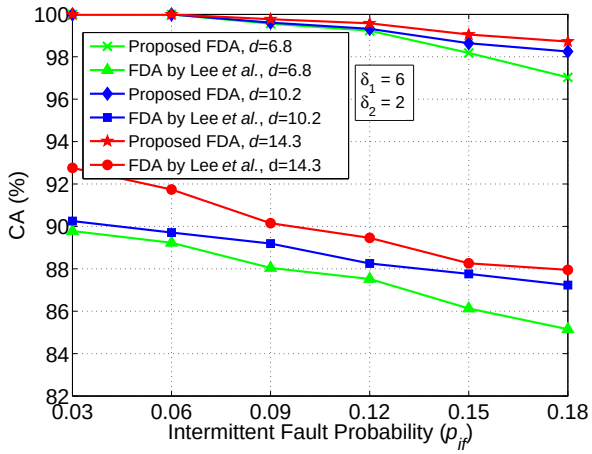
5.3.1 Simulation scenario 1

The first simulation scenario is created for a sensor network with 1000 nodes randomly deployed over $1000 \times 1000\, m^2$ area. With proper adjustment of the transmission range (common for all nodes), the desired value of d can be obtained. In the simulation, the sensor nodes are randomly chosen to have permanently faulty sensors with probabilities 0.02, 0.04, 0.06, 0.08, 0.10, and 0.12 respectively. It is also considered that p_{if} is 150% of p_{pf}, in each case. The values of the thresholds δ_1, and δ_2 are considered 6 and

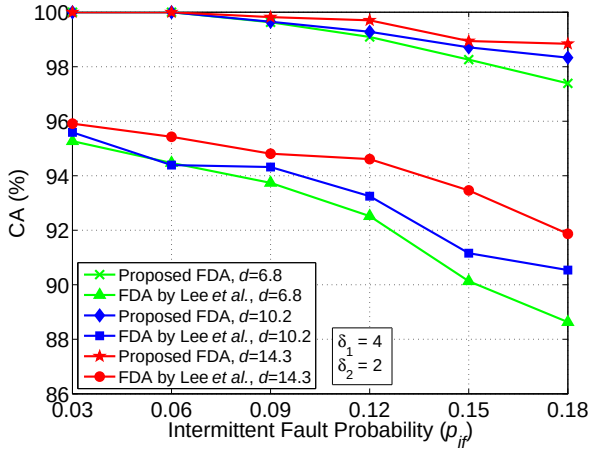
4 respectively. In order to evaluate Lee and Choi's algorithm, the same simulation scenario is considered with $\theta_1 = \lceil d/2 \rceil$, and $(r - \theta_2) \simeq \hat{\theta}_2 = 2$ as the values of the thresholds used in their algorithm. The FDA is run for $r(= 10)$ rounds to handle intermittent faults. The obtained simulation results for CA and FAR are compared as depicted in Figure 5.1(a), 5.2(a), 5.3(a), and 5.4(a).

5.3.2 Simulation scenario 2

The second simulation scenario is created with same fault percentages as in scenario 1. However, the thresholds δ_1, and δ_2 are considered to have values 4 and 2 respectively.

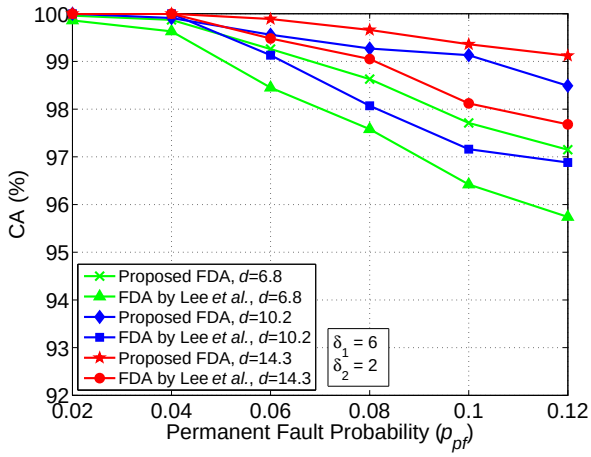


(a)

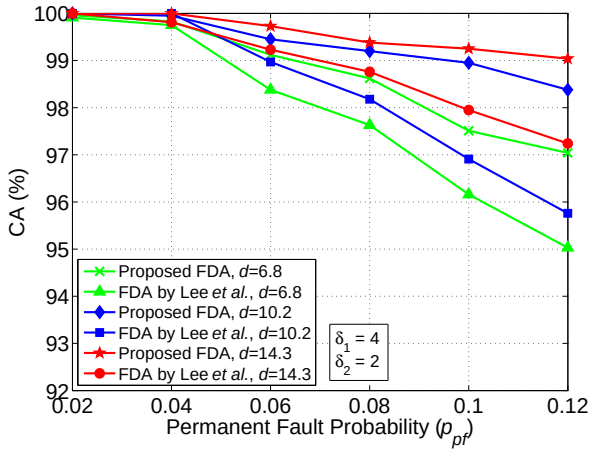


(b)

Figure 5.2: Comparison of intermittent fault CA with (a) δ_1=6, δ_2=2 and (b) δ_1=4, δ_2=2.



(a)



(b)

Figure 5.3: Comparison of fault-free CA with (a) δ_1=6, δ_2=2 and (b) δ_1=4, δ_2=2.

The values of the parameter in Lee and Choi's algorithm are the same as in the first simulation scenario. Figure 5.1(b), 5.2(b), 5.3(b), and 5.4(b) show the obtained results for CA, and FAR in this scenario.

5.3.3 Discussion

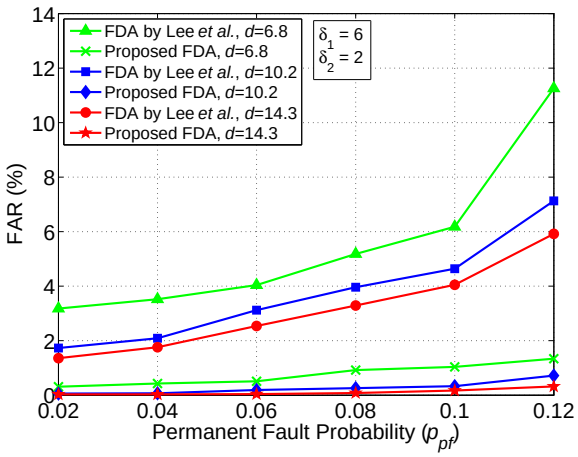
It can clearly be observed that the CA decreases with lower node degrees. This is because the fault-free sensor nodes may not always form a connected graph for fault diagnosis purpose, in case of sparse networks. In such scenarios, all neighbors of a particular node may be faulty at the same time, leading to misdiagnosis of the node. Such scenarios arise with more counts for low d and high fault probability, in which case the performance even degrades.

Figure 5.1 depicts the comparison of classification accuracy for permanently faulty nodes with d values 6.8, 10.2, and 14.3. In some rounds, if a permanently faulty node produces a sensor measurement that does not differ from the sensor measurements of its fault-free neighbors by a minimum threshold δ_1, then it is not classified as permanently faulty. Moreover, the high value of δ_1 boosts the occurrence of such scenarios. The additional threshold test on the residual energy in the proposed FDA handles such cases and improves the performance.

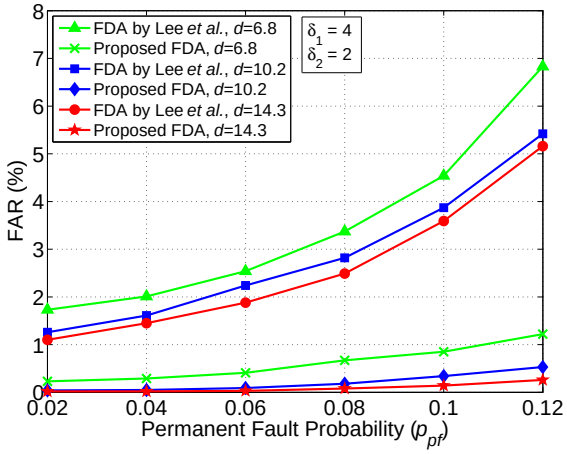
An intermittently faulty node that generates incorrect sensor measurements in less than or equal to θ_2 rounds are not classified as intermittently faulty in the algorithm by Lee and Choi.

Fault-free nodes are diagnosed with better accuracy for high value of δ_1. If the value of δ_1 is chosen to be small, and in some rounds a fault-free node generates sensor measurements that differ from the sensor measurements of its neighboring nodes by the minimum threshold δ_1, then it will not be diagnosed as fault-free by its neighbors. Such miss-classification scenarios are suppressed in the proposed FDA by the additional energy based threshold test.

The comparison of false alarm rates are clearly shown in Figure 5.4. As expected, it is found that with increase in fault probability, FAR increases. Moreover, the FAR is high, for high value of $\delta_1(=6)$ as compared to the low value of $\delta_1(=4)$; with respect



(a)



(b)

Figure 5.4: Comparison of False Alarm Rates with (a) δ_1=6, δ_2=2 and (b) δ_1=4, δ_2=2.

to the same value of δ_2(=2) in both the cases.

The simulation results show that if thresholds are not chosen carefully, then the performance may be the worst. The protocol achieves better performance when average node degree, d is relatively higher, *i.e.*, for denser networks.

5.4 Summary

This chapter presented a distributed fault diagnosis algorithm for WSNs, in order to handle sensor nodes having permanently faulty sensor or intermittently faulty processing unit. The proposed FDA not only diagnoses the faulty sensor nodes, also

classifies them based on their fault types. The algorithm is based on two threshold tests: (i) on sensor measurements of neighboring nodes, and (ii) on residual energy levels of the sensor nodes. Two special cases of intermittent faults are considered: One, where an intermittently faulty node sends similar sensor measurement and similar residual energy level to some neighboring nodes, in all r rounds; Another, where at least one of these values differ in all r rounds. Through mathematical analysis the proposed FDA is proved to be correct and complete. The experimental results show that the algorithm detects and classifies the faulty nodes with high accuracy and low false alarm rate, even in case of high fault probability. Hence, the proposed FDA is proved to be well suited for WSNs. In this work, the transient faults are treated in the same way as intermittent faults. However, the transient faults are manifested due to external errors, and are often considered fault-free. Therefore, it is highly important to discriminate such faults from intermittent types. This is the subject of discussion in next chapter.

Chapter 6

Adaptive Fault Characterization in WSNs

The indispensable objective of a fault diagnosis algorithm is to isolate the nodes that are detected faulty, as early as possible. However, the pragmatic issue that good nodes suffering from transient faults can be detected and treated, but still do not need isolation, is implicitly ruled out by this. Needlessly, isolating such nodes from the network, not only affects the reliability and availability, but also affects the network coverage and sometimes may lead to system break-down. The fault diagnosis algorithm discussed in Chapter 5 suffers from this problem. This chapter presents an online Fault Inspection and Recovery (FIR) approach to study the behavior of transiently as well as intermittently faulty nodes and to segregate them from permanently faulty ones.

The proposed approach is developed on the basis of gathering *fault diagnosis* information over system operations and inspecting the health (persistence and recurrence of failures) of a node to determine, if it needs to be excluded from the network. Therefore, even if some faults are observed in a sensor node, its exclusion from the network is postponed until the fault inspection and subsequent recovery phase confirms its removal.

The subsequent sections of this chapter are organized as follows. The proposed FIR approach, that involves diagnosis, inspection, and recovery, is presented in Section 6.1. In order to vouch the correctness and completeness of the FIR approach, mathematical

analysis is followed in Section 6.2. The performance of the proposed approach is supported by the obtained simulation results as shown in Section 6.3. Finally, a summary of the work is given in Section 6.4.

6.1 Proposed FIR Approach

This section introduces and explores the generic online FIR approach to characterize the faults based on their persistence, and finally reintegrating the nodes affected by transient outages. The proposed FIR approach aims to balance three conflicting demands of a fault-tolerant algorithm.

(i) To exclude, as soon as possible, all sensor nodes with permanent or intermittent faults.

(ii) To avoid miss-signalling any good node as faulty.

(iii) To reintegrate, as soon as possible, all sensor nodes without any fault or possibly with transient faults.

The FIR approach consists of three key phases:

(i) Fault Diagnosis

(ii) Fault Inspection, and

(iii) Fault Recovery

In the *Fault Diagnosis* phase, each node compares its sensor reading and residual energy level with that of its neighbors to decide their fault status. The local view generated at each node is then disseminated to each fault-free node in the network. Once the *Fault Diagnosis* phase deduces a node to be faulty, the *Fault Inspection* phase subsequently monitors the health of the node with respect to the recurrence of faults to decide if node isolation is required. The purpose of this phase is to reintegrate the transiently faulty nodes and isolate the intermittently or permanently faulty ones. However, bursty transients, in case of adverse external conditions, lead to incorrect node isolation in *Fault Inspection* phase. *Fault Recovery* phase aims to handle nodes affected by bursty transients. These phases are discussed in detail, in sequel.

6.1.1 Fault Diagnosis

The WSN is considered to be synchronous, where messages are broadcast and received by the sensor nodes periodically at discrete points in time, at the boundary of each diagnostic round. The diagnosis mechanism is similar to the one discussed in Chapter 5. At the beginning of each round, every node sets a timer to T_{out} and expects to receive messages from its 1-hop neighbors before T_{out} expires. If a neighbor does not respond within T_{out}, then it is diagnosed as faulty. The received sensor readings are analyzed for faults based on the fact that sensor readings are spatially correlated, *i.e.*, sensors covering same region will have sensed similar readings at a particular time. Hence, if $x^t_{S_i}$ and $x^t_{S_j}$ are the sensor readings of two healthy neighboring sensor nodes S_i and S_j respectively, then $|x^t_{S_i} - x^t_{S_j}| < \delta_1$, where δ_1 is an application specific predefined threshold.

Algorithm 6.1: Fault Diagnosis

Data: $C = (S, L)$: Communication graph of WSN.
Result: at each node S_i, the global view about the fault status of all other nodes in the network.
// Node S_i executes this algorithm in each diagnostic round
Activate timer T_{out}
To each node $S_j \in N_{S_i}$, S_i sends own sensor reading, *i.e.*, $x^t_{S_i}$, and residual energy value, *i.e.*, $E^t_{S_i}$
S_i collects $x^t_{S_j}$ and $E^t_{S_j}$ from each neighbor S_j
if T_{out} *expires* **then**
    if S_j *has not responded* **then**
        Set $LocalView_{S_i}[S_j]$ as faulty
    else
        if $|x^t_{S_i} - x^t_{S_j}| < \delta_1$ ***and*** $|E^t_{S_i} - E^t_{S_j}| < \delta_2$ **then**
            Set $LocalView_{S_i}[S_j]$ as fault-free
        else
            Set $LocalView_{S_i}[S_j]$ as faulty
        end
    end
end
Disseminate the Local View to all 1-hop and 2-hop neighbors.
for ***each*** $S_j \in N_{S_i}$ **do**
    Follow majority voting among the decisions of all $S_k \in N_{S_j}$
end
Disseminate the local view over fault-free ST.

To strengthen the diagnosis process, nodes also send their residual energy levels

along with the sensor measurements. The rationale is that in a homogeneous sensor network, the nodes in close proximity have similar levels of residual energy since they do the same duty [59, 101]. Hence, if S_i and S_j are healthy neighboring nodes, then $|E^t_{S_i} - E^t_{S_j}| < \delta_2$, where $E^t_{S_i}$ and $E^t_{S_j}$ are the residual energy levels of S_i and S_j respectively, at time t. Here, δ_2 is a predefined threshold.

Node S_i, to derive more accurate decision about the fault status of neighbor S_j, follows majority voting among the decisions of all neighbors of S_j. The local views are then propagated to all fault-free nodes in the network using spanning tree based dissemination.

The diagnosis protocol ensures, a good node is never diagnosed as faulty and a faulty node is never diagnosed as good (*correctness*), all faulty nodes are diagnosed properly (*completeness*), and all nodes have the same information about the fault status of other nodes in the network in each diagnostic round (*consistency*). The steps followed in *Fault Diagnosis* phase are more precisely depicted in Algorithm 6.1.

6.1.2 Fault Inspection

Once a node is reported faulty, the aim of this phase is to constantly monitor the behavior of the node in subsequent diagnostic rounds based on the correct, complete, and consistent global views provided by the diagnosis protocol. For this purpose, a similar count and threshold algorithm, as adopted in [102], has been used in this work. Two counters: *reward* (r), and *penalty* (p), are used to record the good, and faulty behaviors, respectively, after the first appearance of fault in a sensor node. After a definite number of diagnostic rounds, either r or p exceeds the corresponding predefined threshold value, that eventually helps in deciding whether to isolate the node or not. *Fault Inspection* phase (Algorithm 6.2) uses two predefined thresholds:

- *Penalty counter threshold (P).* The threshold value of penalty counter; if reached, confirms the isolation of an erroneous node.

- *Reward counter threshold (R).* The threshold value of reward counter; if reached, previously suspected erroneous node gets reintegrated into the network as a "good" node.

Algorithm 6.2: Fault Inspection

Data: t_p: number of times node S_j has been penalized since first appearance of fault.
t_r: number of times node S_j has been rewarded since last appearance of fault.
p: penalty counter, r: reward counter.
P: penalty threshold, R: reward threshold.
Result: decision on if S_j is to be ISOLATED or REINTEGRATED.
Initialization: $t_p = t_r = p = r = 0$;
// Node S_i executes this algorithm in each diagnostic round for each neighbor S_j
// after executing fault diagnosis algorithm

```
1  if S_j is faulty then
2  |   p = p + f_p(t_p);
3  |   r = 0;
4  |   t_r = 0;
5  |   t_p = t_p + 1;
6  |   if p ≥ P then
7  |   |   ISOLATE S_j;
8  |   end
9  else
10 |   if p ≠ 0 then
11 |   |   r = r + f_r(t_r);
12 |   |   t_r = t_r + 1;
13 |   |   if r ≥ R then
14 |   |   |   REINTEGRATE S_j;
15 |   |   end
16 |   end
17 end
```

The execution of the *Fault Inspection* algorithm on a single node S_i, is described below.

Initially, all the sensor nodes in the network are good, and the values of both penalty and reward counters, *i.e.*, p and r are set to 0. As long as no error is found in a node, the algorithm doesn't take any action. However, if a node is diagnosed as faulty for the first time, the algorithm inspects the behavior of the node for a finite number of diagnostic rounds before p, or r reaches the respective predefined threshold. In each subsequent rounds, when a node is diagnosed as faulty, the related penalty counter is increased by a *penalty increment*. Conversely, if it is diagnosed as good, then the reward counter is increased by a *reward increment*; but is set to 0 as soon as another fault appears. If the value of the penalty counter exceeds the predefined penalty threshold P, then the node is isolated, and if the value of the reward counter exceeds the predefined threshold R, then the node is reintegrated. In either case, both

the penalty and the reward counters are reset to 0. In order to have faster isolation of faulty nodes, an adaptive increment in penalty counter is followed, instead of a constant increment. In the similar manner an adaptive increment in reward counter is followed to have faster reintegration of the good nodes with transient outages. The algorithm uses two parameters, t_p and t_r, to keep track the number of times a node has been penalized since the first appearance of fault, and rewarded since the last appearance of fault. As the value of t_p increases, the *penalty increment* increases; and, as the value of t_r increases, the *reward increment* increases. Each time a new fault appears in the suspected erroneous node, the value of t_r is reset to 0. The dynamic penalty and reward increments are dependant on the value of t_p and t_r, and are found through two functions f_p and f_r respectively.

Algorithm 6.3: Fault Recovery

Data: t_p: number of times node S_j has been penalized since first appearance of fault.
t_r: number of times node S_j has been rewarded since last appearance of fault.
p: penalty counter, r: reward counter.
P_r: penalty threshold during recovery, R_r: reward threshold during recovery.
Result: decision on if S_j is to be EXCLUDED or REINTEGRATED.
Initialization: $t_p = t_r = p = r = 0$;
// Node S_i executes this algorithm in each diagnostic round for each neighbor S_j,
// in subsequent rounds after the ISOLATION of S_j by *Fault Inspection* Algorithm

```
1  if S_j is faulty then
2  |   p = p + f_p(t_p);
3  |   r = 0;
4  |   t_p = t_p + 1;
5  |   if p ≥ P_r then
6  |   |   EXCLUDE S_j;
7  |   end
8  else
9  |   r = r + f_r(t_r);
10 |   t_r = t_r + 1;
11 |   if r ≥ R_r then
12 |   |   REINTEGRATE S_j with initial penalty p = k;
13 |   end
14 end
```

6.1.3 Fault Recovery

Due to the unfavorable external conditions, the WSNs are deployed in; sensor nodes are likely to be affected by long and bursty transients. In such cases, the *Fault*

Inspection algorithm isolates them from the network. The *Fault Recovery* phase tries to handle such incorrect isolation of the nodes following subsequent observation of node behavior. The *Fault Recovery* algorithm (Algorithm 12) works in a similar way as the *Fault Inspection* algorithm.

During this phase, if p exceeds a predefined *recovery-penalty-threshold* (P_r), then the node is excluded from the network. But if r exceeds a predefined *recovery-reward-threshold* (R_r), then the node was incorrectly isolated due to quick bursty transients and is reintegrated to the network, increasing the availability of good nodes.

The penalty and reward thresholds used in recovery phase may be different from that of inspection phase. During reintegration in recovery phase, a node is assigned a positive penalty k so that successive fault manifestations will lead to faster isolation.

6.2 Analytical Study of Proposed FIR approach

This section concentrates on the mathematical analysis to prove the correctness and completeness of the proposed FIR approach.

6.2.1 Correctness of Proposed FIR approach

The FIR approach is said to be *correct*, if no good node (including nodes with transient faults) is wrongly excluded from the network and no faulty node (permanent or intermittent) is reintegrated back to the network, at the end of *Fault Recovery* phase. The following lemmas are put to support the correctness of the proposed approach.

Lemma 6.1. *In a sensor network, represented by an undirected communication graph $C = (S, L)$; where S and L respectively represents the set of sensor nodes and the set of logical links among them, let $F_P \subset S$, and $F_I \subset S$ respectively be the set of permanently, and intermittently faulty nodes in the network. If $S_j \in (F_P \cup F_I)$, then at the end of the Fault Recovery phase it is never reintegrated back to the network.*

Proof. After a node $S_j \in (F_P \cup F_I)$ has been found faulty for the first time, depending on its faulty or good behavior in subsequent rounds the penalty counter (p), and the reward counter (r) are incremented respectively. The adaptive increment to p and r are decided using two functions, $f_p(t_p)$ and $f_r(t_r)$; where t_p and t_r respectively represent the number of times the node S_j has been penalized or rewarded. The functions are chosen to have minimum value 1.

If S_j is permanently faulty then in *Fault Inspection* phase, p is incremented in each subsequent rounds and, if S_j is intermittently faulty then due to its inherent property of not possessing fault-free behavior for longer period of time, the value of r never exceeds or equals to the predefined reward threshold R. Hence, in either case, after finite number of rounds, the value of p exceeds or become equal to the predefined penalty threshold P; at which point S_j is isolated from the network.

For permanently or intermittently faulty nodes, the *Fault Recovery* algorithm works similar to the *Fault Inspection* algorithm. So after a certain number of rounds, p would be equal to or greater than the predefined penalty threshold for recovery P_r, and the nodes are eventually excluded from the network. Therefore, if S_j is intermittently or permanently faulty, *i.e.*, $S_j \in (F_P \cup F_I)$, then at the end of *Fault Recovery* phase it is excluded from the network. □

Lemma 6.2. *Let S, the set of sensor nodes and L, the set of logical links, collectively define an undirected graph $C = (S, L)$, that represents the communication graph of a sensor network. Let $FF \subseteq S$ be the set of fault-free nodes, and $F_T \subset S$ be the set of transiently faulty nodes in the network. At the end of the Fault Recovery phase, no $S_j \in (FF \cup F_T)$ is excluded from the network.*

Proof. If sensor node S_j is fault-free then it is obvious that it is not excluded from the network, since it is not considered for isolation in *Fault Inspection* phase and hence, never for exclusion in *Fault Recovery* phase. If $S_j \in F_T$ then, after encountering first fault in it, the penalty counter p, and the reward counter r are respectively incremented depending on its faulty or good behavior in the subsequent rounds. By definition, transient faults do not recur themselves so frequently. So the value of r will increase at a faster rate and exceeds or equals to the predefined reward threshold R before value of p exceeds or equals to the penalty threshold P. At this point of *Fault Inspection* phase, node S_j is reintegrated back to the network. However, if S_j suffers from bursty transients then it is isolated from the network in *Fault Inspection* phase. But in the *Fault Recovery* phase, if such isolated nodes exhibit good health condition for a longer period of time, then fast increments in r reintegrates S_j back to the network. Thus, if S_j is fault-free then it is never isolated from the network, and if S_j is transiently faulty then either it is reintegrated back to the network in *Fault Inspection* phase or in *Fault Recovery* phase (in case of bursty transients). □

Theorem 6.1. (*Proof of Correctness*) *In an undirected graph, $C = (S, L)$ that represents the communication graph of a sensor network, where S is the set of sensor nodes and L is the set of logical links between them, let $FF \subseteq S$, $F_P \subset S$, $F_I \subset S$, and $F_T \subset S$ respectively represents the set of fault-free, permanently faulty, intermittently faulty, and transiently faulty nodes in the network. If a node S_j is faulty permanently*

or intermittently, it is always excluded from the network at the end of Fault Recovery phase. However, if S_j is either fault-free or affected by transient faults then it is never excluded from the network.

Proof. The proof of this theorem directly follows Lemma 6.1 and Lemma 6.2. □

6.2.2 Completeness of Proposed FIR approach

The proposed FIR approach is said to be *complete*, if the union of the set of all nodes excluded from the network and the set of all nodes that are part of the network after *Fault Recovery* phase is equivalent to the initial set of nodes in the network, and the proof can be given as in Theorem 6.2.

Theorem 6.2. (*Proof of Completeness*) *Let the communication graph of a sensor network is represented as an undirected graph $C = (S, L)$, where S is the set of sensor nodes and L is the set of logical links between them. If FF is the set of fault-free nodes, then $(FF \cup EX \cup RE) = S$ holds; where EX, and RE are the set of nodes excluded, and reintegrated respectively at the end of Fault Recovery phase.*

Proof. Lemma 6.1 conveys that if a node $S_j \in (F_P \cup F_I)$; where F_P is the set of permanently faulty nodes and F_I is the set of intermittently faulty nodes, then in the *Fault Inspection* phase it is isolated and subsequently excluded from the network in *Fault Recovery* phase and hence, $S_j \in EX$. From Lemma 6.2, it can be followed that if a node $S_j \in FF$ then it is never isolated in the *Fault Inspection* phase and hence, abstained from exclusion in *Fault Recovery* phase. If $S_j \in F_T$; where F_T is the set of transiently faulty nodes in the network, then it is reintegrated back to the network in the *Fault Inspection* phase itself. However, if it is the case of bursty transients then despite of its isolation in *Fault Inspection* phase it is reintegrated back to the network based on its fault-free behavior in subsequent rounds in *Fault Recovery* phase. Thus, if $S_j \in F_T$ initially, then $S_j \in RE$ at the end of *Fault Recovery* phase. So, any node $S_j \in S$, is either a part of FF or included in EX or RE at the end of *Fault Recovery* phase and hence, $(FF \cup EX \cup RE) = S$ holds. □

Theorem 6.3. *The communication overhead of the proposed protocol is $O(nd)$, where n is the number of nodes in the network and d is the average node degree.*

Proof. The protocol proceeds in three phases: fault diagnosis, fault inspection, and fault recovery phase. The fault diagnosis algorithm is same as the one discussed in Chapter 5, and the message complexity is shown to be $O(nd)$ [Theorem 5.3]. In fault inspection and subsequent recovery phase, a node takes the results of the fault

diagnosis to simply decide if a node is to be excluded or to be reintegrated back to the network. So the message complexity of the proposed protocol is the complexity of the the fault diagnosis algorithm itself. Thus, the communication complexity of the proposed protocol is $O(nd)$. □

6.3 Simulation and Results

The performance of the proposed FIR scheme is evaluated through simulation using the Castalia-3.2 simulator on OMNeT++ 4.2 platform. This section presents the comparative results obtained for the proposed approach with respect to the scheme discussed by Serafini *et al.* [102]. The objective of the proposed scheme is to exclude the faulty nodes (nodes affected by permanent, or intermittent faults) from the network, and reintegrate the transiently faulty ones (considering them to be good) back to the network, as early as possible. Keeping this objective in place, the following four performance evaluation metrics are used for discussion:

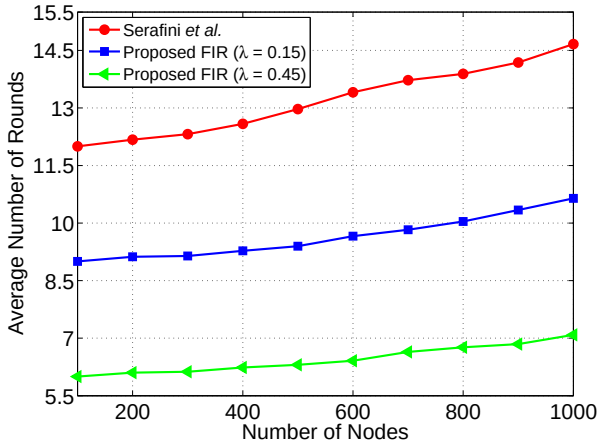
- Average number of Rounds for Exclusion (ARE),
- Average number of Rounds for Reintegration (ARR),
- Detection Accuracy (DA), and
- False Alarm Rate (FAR).

Former two metrics are self explanatory and hence, need no further description. DA is defined as the ratio of the combination of the number of faulty nodes excluded, the number of transiently faulty nodes reintegrated, and the number of fault-free nodes diagnosed fault-free to the total number of nodes in the network. FAR is the ratio of the sum of the number of faulty nodes reintegrated and the number of fault-free or transiently faulty nodes excluded to the total number of nodes in the network.

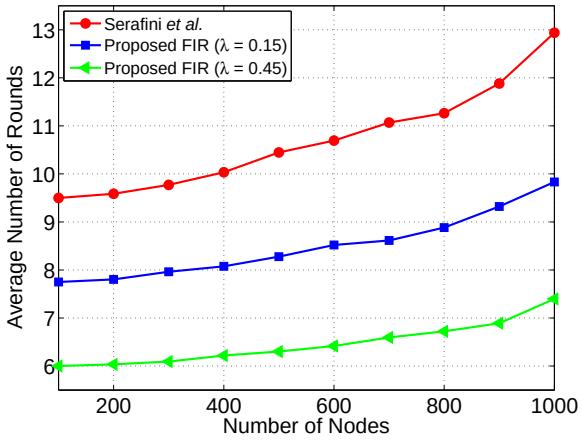
Two functions, f_p and f_r are used in Algorithm 6.1 and 6.2 to decide the adaptive increments to the penalty and reward counters respectively. Without loss of generality, f_p and f_r are considered to be the same, and we use the exponentiation function for the purpose. Two different values of λ: 0.15 and 0.45 are considered. The penalty threshold P, and the reward threshold R are considered to have the same value 5.

The *Fault Inspection* and *Fault Recovery* phase are repeated for twelve rounds each, for monitoring the health of the nodes after the first fault manifestation in them.

Different simulation scenarios are created to adjudge the effectiveness of the proposed scheme.



(a)



(b)

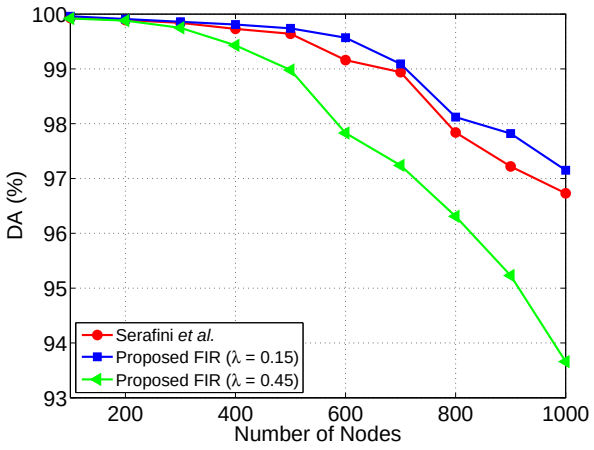
Figure 6.1: Comparison of average number of rounds for (a) exclusion, and (b) reintegration, with varying network size.

6.3.1 Simulation scenario 1

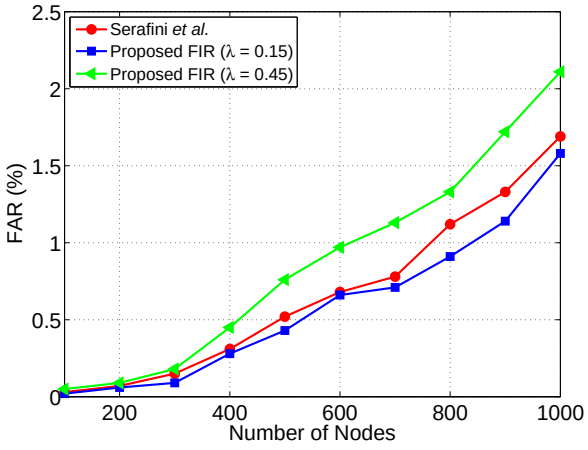
The first simulation scenario is set up with 6% permanently faulty nodes (faulty in all rounds) and 6% intermittently faulty nodes (faulty in alternate rounds). Two specific cases of transient faults are considered: 6% of total nodes showing transient faults

in only one round (*type-1*), and another 6% showing transient faults in two rounds (*type-2*). 6% of nodes are affected by bursty transients until they are isolated from the network in the *Fault Inspection* phase. However, in the *Fault Recovery* phase due to their fault-free or transient behavior (either type-1 or type-2 with equal probability), they are reintegrated back to the network.

Considering the above mentioned fault classes and the adaptive increment functions, the obtained simulation results for ARE, ARR, DA, and FAR for varied network size (100 through 1000 at an increment of 100), eventually for varied average node degree[2], are depicted in Figure 6.1, and 6.2.



(a)



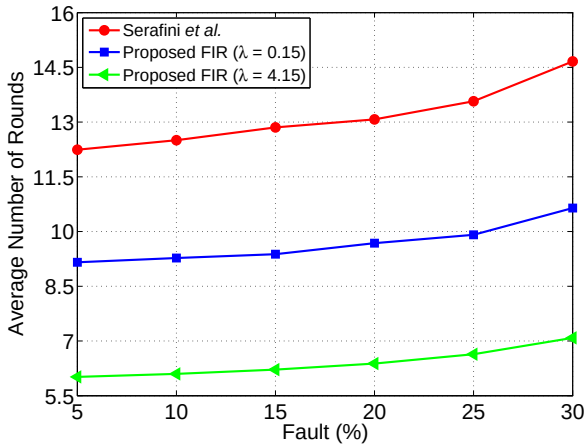
(b)

Figure 6.2: Comparison of (a) DA, and (b) FAR, with varying network size.

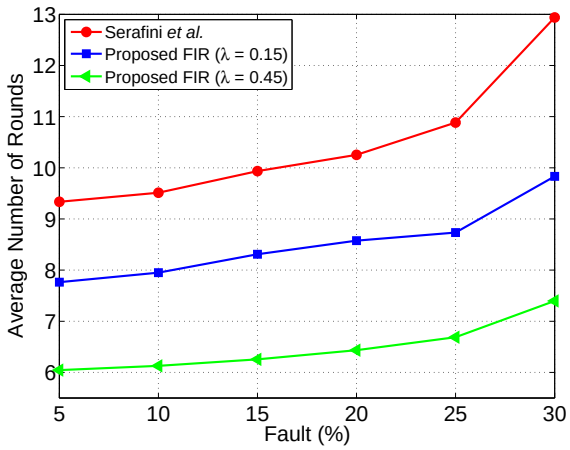
[2]Keeping the deployment area to be same, varying the number of nodes in the network proportionally varies the average node degree in the network.

6.3.2 Simulation scenario 2

Another simulation scenario is created with 1000 sensor nodes deployed over a 1000×1000 m^2 area to study the behavior of the aforementioned performance evaluation metrics with varied fault percentage. We consider the total fault percentage from 5% to a maximum of 30% in the network. The percentage of various fault classes, as mentioned in Scenario 1, are in equal proportion. The obtained results for the existing approach by Serafini *et al.*, and the proposed FIR approach for $\lambda = 0.15$ and $\lambda = 0.45$ are compared clearly in Figure 6.3, and 6.4.



(a)



(b)

Figure 6.3: Comparison of average number of rounds for (a) exclusion, and (b) reintegration, with varying fault %.

6.3.3 Discussion

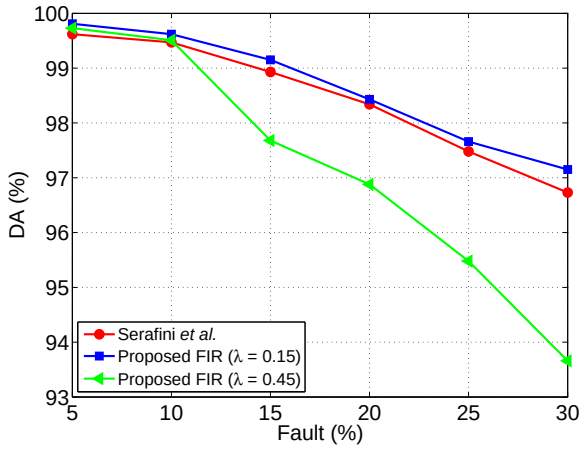
Figure 6.1 and 6.3 conveys that ARE and ARR decreases, as expected, with high value of λ. On the contrary, from Figure 6.2 and 6.4, it is clear that the high value of λ leads to high FAR with less DA. This shows a clear trade-off.

The accuracy of the FIR approach is dependent on the consistent global view generated at each sensor node, at the end of each round of *Fault Diagnosis* (Algorithm 6.1). However, the inconsistency in the global view increases with the increased network size, due to high packet loss during dissemination of local diagnostics (Line 15 and 19, Algorithm 6.1). This results in increasing FAR and decreasing DA, which in turn affects the average number of rounds required for exclusion and reintegration negatively.

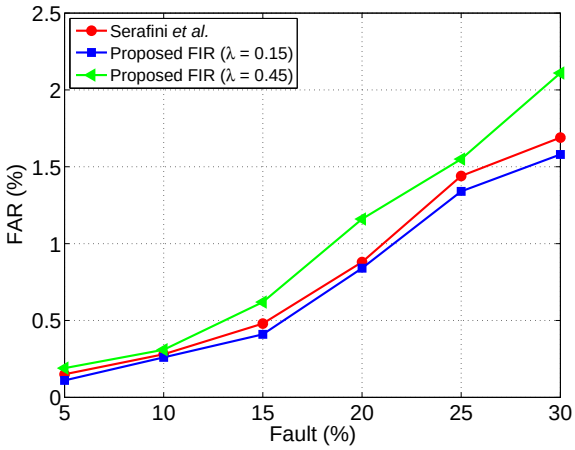
More number of faulty nodes may sensitively affect the decision in the majority voting protocol in Fault Diagnosis phase. It may so happen that the subgraph overlaying all fault-free nodes in the network may not form a connected graph for fault diagnosis. High fault percentage further aggravates these problems. This not only affects the DA, and FAR of the diagnosis but also affects the ARR and ARE.

The reasons for the degraded performance of the proposed approach with respect to DA and FAR when $\lambda = 0.45$ are twofold: (i) A node affected by bursty transients is isolated in the *Fault Inspection* phase. However, if it shows transiently faulty (type-2) behavior in the *Fault Recovery* phase, and misdiagnosed as faulty in some early round, then it is excluded from the network. Such scenarios arise more in number in case of large networks and high fault rate. (ii) If a node is transiently faulty, then it may be wrongly isolated from the network in *Fault Inspection* phase as a side effect of misdiagnosis in some early round(s). The node may be excluded from the network as a consequence of misdiagnosis in *Fault Recovery* phase.

At low value of λ, *i.e.*, 0.15, the obtained DA and FAR along with ARE and ARR are better as compared to that of the existing approach by Serafini *et al.*. In case of transiently faulty nodes, the existing approach must correctly diagnose the fault-free behavior of the nodes for R (=5 in this scenario) consecutive rounds before taking a decision about the reintegration of the node to the network. In case, the node



(a)



(b)

Figure 6.4: Comparison of (a) DA, and (b) FAR, with varying fault %.

is misdiagnosed as faulty, again it has to wait for R consecutive correct decisions about the fault-free status of the node. Furthermore, the additional residual energy level comparison in *Fault Diagnosis* phase of the proposed algorithm improves the diagnosis.

6.4 Summary

An adaptive Fault Inspection and Recovery approach to characterize and hence discriminate transient faults from intermittent, and permanent faults in WSNs is the key discussion in this chapter. After a node is detected as faulty in *Fault Diagnosis* phase for the first time, the health of the node is inspected in subsequent rounds. Two

counters, namely, penalty counter (p) and reward counter (r) are incremented based on the faulty or good behavior of the node respectively. The node is considered to be transiently faulty and reintegrated back to the network, if r reaches a predefined threshold R. However, if p reaches a predefined threshold P early, then the node is excluded from the network considering it to be intermittently or permanently faulty. The implication of the *Fault Recovery* phase is to reintegrate back the nodes those are isolated in the *Fault Inspection* phase due to bursty transients. The adaptive increment to p and r leads to faster reintegration or exclusion. The simulation results show that the proposed FIR approach outperforms the existing approach by Serafini *et al.* with respect to high detection accuracy and low false alarm rate, even in case of larger network and high fault probability, by properly tuning the parameter λ in the adaptive increment function.

Chapter 7

Conclusions and Future Work

Faults are inevitable in wireless networks due to the uncooperative operational environment in which they are deployed, or hardware and/or software related flaws in the nodes of the network. Presence of some faulty nodes does not conclude complete breakdown of the network, although reduces the efficiency and affects the quality of service (QoS). Application of wireless networks in numerous critical applications demands for the development of efficient FDPs. Drawing a consensus among the fault-free units about the status of all faulty units in the network, is the underlying objective of a FDP. This thesis work unravels along two directions: designing FDPs for MANETs, and for WSNs.

FDPs for MANETs are developed based on the *comparison model*, *i.e.*, by comparing the results of executions of same test tasks by different nodes. In other hand, *spatial correlation* between the measurements of sensor nodes forms the basis of diagnosis in WSNs. The spatial correlation property states that the difference between the measurements of any pair of fault-free neighboring sensor nodes do not exceed a particular threshold. However, if one of those is faulty, then the difference is significant. An additional consideration that the nodes in the same proximity, in homogeneous WSNs, have same residual energy value; strengthens the diagnosis. Through analytical study, the proposed algorithms are shown to generate *correct* and *complete* diagnosis view at each fault-free node in the network. After each node in the network finds the *local view* containing the fault status of all 1-hop neighbors, disseminates the same to all other nodes so that each fault-free node can deduce the *global view* of the

network. Spanning tree based dissemination is adopted to reduce the communication complexity. The obtained simulation results also vouch the feasibility and efficiency of the proposed approaches.

Considering the static topology of the MANET, a distributed FDP is devised to handle permanent faults (hard and soft). Each node responds to only one test request in the proposed protocol, which reduces the message communication compared to *static*, *dynamic*, and *adaptive* diagnosis protocols [8, 10, 32]. Static protocol requires a node to respond to all incoming test requests, and in dynamic, and adaptive protocols, a node responds to $(\sigma+1)$ test requests, where σ is the diagnosability of the network. The DA and FAR of dynamic, adaptive, and proposed diagnosis protocol are comparable, but static protocol results better DA, and FAR because of the ability to cope with the diagnostic information loss during, flooding based, dissemination. However, flooding based dissemination suffers from the message implosion problem, and results in high communication overhead. The constraint on the node mobility is unleashed and a novel diagnosis protocol is proposed to handle more intractable intermittent faults in MANETs. Time redundancy is used to handle such faults. The obtained results clearly show the deleterious effect of mobility on DA and FAR.

Handling intermittent faults, and classifying the nodes based on their fault types in more error-prone WSNs is also discussed. Two special cases of intermittent faults are considered: one, where an intermittently faulty node sends similar sensor measurement and similar residual energy level to some neighboring nodes, in all r rounds; another, where at least one of these values differ in all r rounds. The performance of the protocols is observed to be better compared to Lee and Choi's fault detection algorithm [62]. Transient faults are often caused due to external errors (e.g., noise), and their adverse effects disappear rapidly. Therefore, nodes affected by such faults are usually considered fault-free, and should not be excluded from the network. In conjunction with this objective, a count-and-threshold based algorithm is proposed to discriminate transient faults from intermittent, and permanent faults. The health of a node, after detected faulty, is monitored further before confirming its exclusion from the network. If found transiently faulty, then the node is reintegrated back to

the network, improving the availability, and computational efficiency. The performance of this protocol is compared with that of Serafini *et. al.* [102], with respect to the average number of rounds required to reintegrate the transiently faulty nodes, to exclude the permanently and intermittently nodes, DA, and FAR.

As reported in the literature, the energy consumption during communication includes energy expended for transmission and reception of messages [58, 103], and thus the energy consumption of any node is directly proportional to the amount of traffic it receives and generates [8]. Hence, a network would experience an extended lifetime with reduction in number of messages required in diagnosis. In our protocols, adoption of ST based dissemination strategy significantly reduces the communication overhead compared to the protocols in [8, 10, 11, 32, 62, 102] where broadcasting based dissemination is followed.

Keen towards the development of *best* protocols, although we always come up with *better* ones, will always increase the thirst towards future research. The performances of the proposed protocols are proven to be superior compared to the state-of-the-art approaches. However, based on the findings thorough this research, there are many ancillary research directions still open, and need further investigation. The fault diagnosis algorithm for static topology WSNs can be extended over to time-varying topology WSNs. In all the works, we have considered there was no communication fault, or if a communication fault arises then the node associated with it is treated faulty. Further investigation is required to detect communication faults, and segregate those from node faults. The faults are considered static, in this thesis work. Handling faults that are induced during diagnosis session, *i.e.*, dynamic faults, would be promising.

Bibliography

[1] R. Isermann and P. Balle. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, 1997.

[2] A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and its threats: A taxonomy. In *Building the Information Society*, volume 156 of *IFIP International Federation for Information Processing*, pages 91–120. Springer US, 2004.

[3] M. Barborak, A. Dahbura, and M. Malek. The consensus problem in fault-tolerant computing. *ACM Computing Survey*, 25(2):171–220, 1993.

[4] A. Bondavalli, S. Chiaradonna, F.D. Giandomenico, and F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *IEEE Transactions on Computers*, 49(3):230–245, 2000.

[5] R.D. Schlichting and F.B. Schneider. Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transaction on Computing Systems*, 1(3):222–238, 1983.

[6] F. Cristian, H. Aghili, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. *Information and Computation*, 118(1):158–179, 1995.

[7] L.A. Laranjeira, M. Malek, and R. Jenevein. On tolerating faults in naturally redundant algorithms. In *Proceedings of 10th Symposium on Reliable Distributed Systems*, pages 118–127, 1991.

[8] M. Elhadef, A. Boukerche, and H. Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 68(3):321–335, 2008.

[9] F. Barsi, F. Grandoni, and P. Maestrini. A theory of diagnosability of digital systems. *IEEE Transactions on Computers*, C-25(6):585–593, 1976.

[10] S. Chessa and P. Santi. Comparison-based system-level fault diagnosis in ad hoc networks. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, pages 257–266, 2001.

[11] S. Chessa and P. Santi. Crash faults identification in wireless sensor networks. *Computer Communications*, 25(14):1273–1282, 2002.

[12] W.A. Arbaugh, N. Shankar, Y.C.J. Wan, and K. Zhang. Your 802.11 wireless network has no clothes. *IEEE Wireless Communications*, 9(6):44–51, 2002.

[13] M.N. Lima, A.L.D. Santos, and G. Pujolle. A survey of survivability in mobile ad hoc networks. *IEEE Communication Surveys & Tutorials*, 11(1):66–77, 2009.

[14] F. Adelstein, S.K.S. Gupta, G.G. Richard III, and L. Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hill, 1 edition, 2005.

[15] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[16] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[17] S.-J. Yim and Y.-H. Choi. An adaptive fault-tolerant event detection scheme for wireless sensor networks. *Sensors*, 10(3):2332–2347, 2010.

[18] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, 2004.

[19] F.P. Preparata, G. Metze, and R.T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computers*, EC-16(6):848–854, 1967.

[20] J. Xu and L. Lilien. A survey of methods for system-level fault diagnosis. In *Proceedings of the Fall Joint Computer Conference on Exploring Technology: today and tomorrow*, ACM'87, pages 534–540, 1987.

[21] A.K. Somani. System level diagnosis: A review. Technical report, Iowa State University, 1997.

[22] S.L. Hakimi and A.T. Amin. Characterization of connection assignment of diagnosable systems. *IEEE Transactions on Computers*, C-23(1):86–88, 1974.

[23] J.G. Kuhl and S.M. Reddy. Distributed fault-tolerance for large multiprocessor systems. In *Proceedings of the 7th annual symposium on Computer Architecture*, ISCA'80, pages 23–30, 1980.

[24] A.K. Somani, V.K. Agarwal, and D. Avis. A generalized theory for system level diagnosis. *IEEE Transactions on Computers*, C-36(5):538–546, 1987.

[25] M. Malek. A comparison connection assignment for diagnosis of multiprocessor systems. In *Proceedings of the 7th annual symposium on Computer Architecture*, ISCA'80, pages 31–36, 1980.

[26] K. Chwa and S. L. Hakimi. Schemes for fault-tolerant computing: A comparison of modularly redundant and t-diagnosable systems. *Information and Control*, 49(3):212–238, 1981.

[27] A.T. Dahbura and G.M. Masson. An $O(n^{2.5})$ fault identification algorithm for diagnosable systems. *IEEE Transaction on Computers*, 33(6):486–492, 1984.

[28] S. Rangarajan, D. Fussell, and M. Malek. Built-in testing of integrated circuit wafers. *IEEE Transaction on Computers*, 39(2):195–205, 1990.

[29] D.M. Blough and A. Pelc. Complexity of fault diagnosis in comparison models. *IEEE Transactions on Computers*, 41(3):318–324, 1992.

[30] Y. Chen, W. Bucken, and K. Echtle. Efficient algorithms for system diagnosis with both processor and comparator faults. *IEEE Transaction on Parallel Distributed Systems*, 4(4):371–381, 1993.

[31] C.P. Fuhrman and H.J. Nussbaumer. Comparison diagnosis in large multiprocessor systems. In *Asian Test Symposium*, pages 244–249, 1996.

[32] M. Elhadef, A. Boukerche, and H. Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, MobiWac'06, pages 18–27, 2006.

[33] A. Weber, A.R. Kutzke, and S. Chessa. Diagnosability evaluation for a system-level diagnosis algorithm for wireless sensor networks. In *IEEE Symposium on Computers and Communications*, pages 241–244, 2010.

[34] A.Weber, A. Kutzke, and S.Chessa. Energy-aware test connection assignment for the self-diagnosis of a wireless sensor network. *Journal of the Brazilian Computer Society*, 18(1):1–9, 2012.

[35] C. Jaikaeo, C. Srisathapornphat, and C.-C. Shen. Diagnosis of sensor networks. In *Proceedings of the IEEE International Conference on Communications*, pages 1627–1632, 2001.

[36] G. Gupta and M. Younis. Fault-tolerant clustering of wireless sensor networks. In *Proceedings of the IEEE International Conference on Wireless Communications and Networking*, pages 1579–1584, 2003.

[37] A.T. Tai, K.S. Tso, and W.H. Sanders. Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 805–814, 2004.

[38] A.F. Iskander and A.A. Younis. A proactive fault tolerance management algorithm for mobile ad hoc networks. In *Proceedings of the 4th IEEE International Conference Consumer Communications and Networking*, pages 571–575, 2007.

[39] O. Younis, S. Fahmy, and P. Santi. An architecture for robust sensor network communications. *International Journal of Distributed Sensor Networks*, 1(3–4):305–327, 2005.

[40] S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 575–584, 2006.

[41] D. Li. Cluster-based system-level fault diagnosis in hierarchical ad-hoc networks. In *Proceedings of the International Conference on Computational Intelligence and Security*, pages 1062–1066, 2007.

[42] P. Wang, J. Zheng, and C. Li. An agreement-based fault detection mechanism for under water sensor networks. In *Proceedings of the IEEE International Conference on Global Telecommunications*, pages 1195–1200, 2007.

[43] G. Venkataraman, S. Emmanuel, and S. Thambipillai. Energy-efficient cluster-based scheme for failure management in sensor networks. *IET Communications*, 2(4):528–537, 2008.

[44] M. Asim, H. Mokhtar, and M. Merabti. A fault management architecture for wireless sensor network. In *Proceedings of the International Conferece on Wireless Communications and Mobile Computing*, pages 779–785, 2008.

[45] X. Li, C. Men, Z. Xu, Z. He, S. Cai, and N. Yao. A fault detection service for cluster-based ad hoc network. In *Proceedings of the 5th International Conference on Internet Computing for Science and Engineering*, pages 171–175, 2010.

[46] L. Gheorghe, R. Rughinis, R. Deaconescu, and N. Tapus. Adaptive trust management protocol based on fault detection for wireless sensor networks. In *Proceeding of the Second International Conference on Advanced Service Computing*, pages 216–221, 2010.

[47] M.Z. Khan, M. Merabti, B. Askwith, and F. Bouhafs. A fault-tolerant network management architecture for wireless sensor networks. In *Proceedings of 11th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, 2010.

[48] A.A. Taleb, J. Mathew, and D.K. Pradhan. Fault diagnosis in multi layered de bruijn based architectures for sensor networks. In *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications*, pages 456–461, 2010.

[49] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 1–9, 2000.

[50] R. Huang, X. Qiu, and L. Rui. Simple random sampling-based probe station selection for fault detection in wireless sensor networks. *Sensors*, 11(3):3117–3134, 2011.

[51] W. Wang, B. Wang, Z. Liu, and L. Guo. A cluster-based real-time fault diagnosis aggregation algorithm for wireless sensor networks. *Information Technology Journal*, 10(1):80–88, 2011.

[52] K. Sakib. Asynchronous failed sensor node detection method for sensor networks. *International Journal of Network Management*, 22(1):27–49, 2011.

[53] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom'00, pages 255–265, 2000.

[54] A. Patcha and A. Mishra. Collaborative security architecture for black hole attack prevention in mobile ad hoc networks. In *Proceedings of the International Conference on Radio and Wireless*, pages 75–78, 2003.

[55] N. Nasser and Y. Chen. Enhanced intrusion detection system for discovering malicious nodes in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Communications*, pages 1154–1159, 2007.

[56] M.C. Vuran, O.B. Akan, and I.F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[57] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250, 2004.

[58] X. Luo, M. Dong, and Y. Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70, 2006.

[59] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, pages 902–913 vol. 2, 2005.

[60] J. Chen, S. Kher, and A. Somani. Distributed fault detection of wireless sensor networks. In *Proceedings of the Workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72, 2006.

[61] C. Hsin and M. Liu. Self-monitoring of wireless sensor networks. *Computer Communications*, 29(4):462–476, 2006.

[62] M.-H. Lee and Y.-H. Choi. Fault detection of wireless sensor networks. *Computer Communications*, 31(14):3469–3475, 2008.

[63] P. Jiang. A new method for node fault detection in wireless sensor networks. *Sensors*, 9(2):1282–1294, 2009.

[64] J.-Y. Choi, S.-J. Yim, Y.J. Huh, and Y.-H. Choi. A distributed adaptive scheme for detecting faults in wireless sensor networks. *WSEAS Transactions on Communications*, 8(2):269–278, 2009.

[65] X. Miao, K. Liu, Y. He, Y. Liu, and D. Papadias. Agnostic diagnosis: Discovering silent failures in wireless sensor networks. In *Proceedings of the 30th Annual IEEE International Conference on Computer Communications*, INFOCOM'11, pages 1548–1556, 2011.

[66] T. Sun, L.-J. Chen, C.-C. Han, and M. Gerla. Reliable sensor networks for planet exploration. In *Proceedings of the IEEE International Conference on Networking, Sensing, and Control*, pages 816–821, 2005.

[67] X.-Y.Xiao, W.-C. Peng, C.-C. Hung, and W.-C. Lee. Using sensorranks for in-network detection of faulty readings in wireless sensor networks. In *Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access*, pages 1–8, 2007.

[68] J. Gao, Y. Xu, and X. Li. Weighted-median based distributed fault detection for wireless sensor networks. *Journal of Software*, 18(5):1208–1217, 2007.

[69] S. Guo, Z. Zhong, and T. He. Find: faulty node detection for wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 253–266, 2009.

[70] C. Zhang, J. Ren, C. Gao, Z. Yan, and L. Li. Sensor fault detection in wireless sensor networks. In *Proceedings of the IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009)*, pages 66–69, 2009.

[71] D. Rakhmatov and S.B.K. Vrudhula. Time-to-failure estimation for batteries in portable electronic systems. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 88–91, 2001.

[72] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi. A discrete-time battery model for high-level power estimation. In *Proceedings of the International Conference on Design, automation and test*, pages 35–41, 2000.

[73] S. Harte, A. Rahman, and K.M. Razeeb. Fault tolerance in sensor networks using self-diagnosing sensor nodes. In *Proceedings of the IEE International Workshop on Intelligent Environments*, pages 7–12, 2005.

[74] M. Sridharan, S. Bapat, R. Ramnath, and A. Arora. Implementing an autonomic architecture for fault-tolerance in a wireless sensor network testbed for at-scale experimentation. In *Proceedings of the ACM symposium on Applied computing*, SAC'08, pages 1670–1676, 2008.

[75] A. Arora and M. Theimer. On modeling and tolerating incorrect software. *Journal of High Speed Networks*, 14(2):109–134, 2005.

[76] Z. Ji, W. Bing-shu, M. Yong-guang, Z. Rong-hua, and D. Jian. Fault diagnosis of sensor network using information fusion defined on different reference sets. In *Proceedings of the International Conference on Radar*, pages 1–5, 2006.

[77] A. Jabbari, R. Jedermann, and W. Lang. Application of computational intelligence for sensor fault detection and isolation. In *Proceedings of the World Academy of Science, Engineering and Technology*, pages 265–270, 2007.

[78] A.I. Moustapha and R.R. Selmic. Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. *IEEE Transactions on Instrumentation and Measurement*, 57(5):981–988, 2008.

[79] D. Zhu, J. Bai, and S.X. Yang. A multi-fault diagnosis method for sensor systems based on principle component analysis. *Sensors*, 10(1):241–253, 2010.

[80] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 1–10, 2000.

[81] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom'99, pages 174–185, 1999.

[82] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom'99, pages 151–162, 1999.

[83] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, 52(5):545–557, 2003.

[84] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(34):353–363, 2001.

[85] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks, INRIA report, March 2000.

[86] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, HICSS'02, pages 3866–3875, 2002.

[87] E. Baccelli, J.A. Cordero, and P. Jacquet. Multi-point relaying techniques with OSPF on ad hoc networks. In *Proceedings of the 4th International Conference on Systems and Networks Communications*, ICSNC'09, pages 53–62, 2009.

[88] E. Baccelli, P. Jacquet, and T. Clausen. OSPF multipoint relay (MPR) extension for ad hoc networks, RFC 5449, 2009.

[89] E. Baccelli, J.A. Cordero, and P. Jacquet. OSPF over multi-hop ad hoc wireless communications. *International Journal of Computer Networks and Communications*, 2(5):37–56, 2010.

[90] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF), RFC 3684, 2004.

[91] R. Ogier and P. Spagnolo. Mobile ad hoc network (MANET) extension of OSPF using connected dominating set (CDS) flooding, RFC 5614, 2009.

[92] E. Baccelli, J.A. Cordero, and P. Jacquet. Optimization of critical data synchronization via link overlay RNG in mobile ad hoc networks. In *Proceedings of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 402–411, 2010.

[93] J.A. Cordero, P. Jacquet, and E. Baccelli. Impact of jitter-based techniques on flooding over wireless ad hoc networks: Model and analysis. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications*, INFOCOM'12, pages 2059–2067, March 2012.

[94] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.P. Vasseur, and R. Alexander. RPL: IPv6 routing protocol for low-power and lossy networks, RFC 6550, March 2012.

[95] Y. Yi and M. Gerla. Efficient flooding in ad hoc networks: a comparative performance study. In *Proceedings of the IEEE International Conference on Communications*, ICC'03, pages 1059–1063, 2003.

[96] B. Parhami. Voting networks. *IEEE Transactions on Reliability*, 40(3):380–394, 1991.

[97] B. Parhami. Voting algorithms. *IEEE Transactions on Reliability*, 43(4):617–629, 1994.

[98] M. Diop, C. Pham, and O. Thiare. 2-hop neighborhood information for cover set selection in mission-critical surveillance with wireless image sensor networks. In *Proceedings of the Wireless Days Conference*, pages 1–7, Nov 2013.

[99] I. Krontiris, Z. Benenson, T. Giannetsos, F. Freiling, and T. Dimitriou. Cooperative intrusion detection in wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, LNCS, pages 263–278, 2009.

[100] C.J. Mallery, S. Medidi, and M. Medidi. Relative localization with 2-hop neighborhood. In *Proceedings of the International Symposium on a World of Wireless, Mobile, and Multimedia Networks*, pages 1–4, 2008.

[101] Y.J. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *Proceedings of the IEEE International Conference on Wireless Communications and Networking*, pages 356–362, 2002.

[102] M. Serafini, A. Bondavalli, and N. Suri. On-line diagnosis and recovery: On the choice and impact of tuning parameters. *IEEE Transactions on Dependable and Secure Computing*, 4(4):295–312, 2007.

[103] J. Martyna. Modeling of energy consumption for mobile wireless ad hoc and sensor networks. In *Proceedings of the International Conference on Computer Networks*, pages 314–323, 2012.

Dissemination

Accepted/Published

1. **Manmath Narayan Sahoo**, and Pabitra Mohan Khilar. Diagnosis of Wireless Sensor Networks in Presence of Permanent and Intermittent Faults. *Wireless Personal Communications, Springer*, 78(2): 1571-1591, 2014. DOI: 10.1007/s11277-014-1836-6.

2. **Manmath Narayan Sahoo**, and Pabitra Mohan Khilar. System-Level Fault Diagnosis in Fixed Topology Mobile Ad hoc Networks. *International Journal of Communication Networks and Distributed Systems, Inderscience*, 10(3): 216-232, 2013. DOI: 10.1504/IJCNDS.2013.053078.

3. **Manmath Narayan Sahoo**, and Pabitra Mohan Khilar. Intermittent Fault Diagnosis in Dynamic Topology MANETs. *International Journal of Signal and Imaging Systems Engineering, Inderscience*, 2014.

Communicated

1. **Manmath Narayan Sahoo**, and Pabitra Mohan Khilar. Adaptive Characterization of Faults Based on their Persistence in Wireless Sensor Networks. *Wireless Sensor Systems, IET*, 2014.

Resume

Manmath Narayan Sahoo

Permanent Address

At–Nagabasta, Po–Ratilo, Nimapara,
Dist-Puri, Odisha – 752 114, India

✉ sahoom[at]nitrkl[dot]ac[dot]in,
sahoo[dot]manmath[at]gmail[dot]com

(+91) 661-246-2364(O),
(+91) 661-246-3364(R),
(+91) 9861222287(M)

Affiliation and Correspondence Address

Assistant Professor,
Department of Computer Science and Engineering,
National Institute of Technology Rourkela,
Odisha – 769 008, India.

Date of Birth: 05 February 1984

Area of Interest

- Fault Tolerance
- Operating Systems
- Distributed Computing

Academic Qualifications

- M.Tech. (Computer Science & Engineering)
  National Institute of Technology, Rourkela, [First division]
- B.Tech (Computer Science & Engineering)
  Kalinga Institute of Industrial Technology, Bhubaneswar, [First division]
- Higher Secondary (+2)
  Council of Higher Secondary Education, Odisha, [First division]
- Madhyamik Examination (10^{th})
  Board of Secondary Education, Odisha, [First division]

Professional Experience

- Assistant Professor, National Institute of Technology, Rourkela, Jul 2009 – Date
- Lecturer, Institute of Technical Education and Research, Bhubaneswar, Nov 2005 – Jul 2007

Publications

- Accepted/Published: Journal (04), Conference(05)
- Communicated: Journal (01), Conference (02)